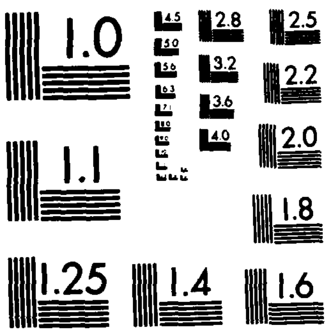
AD-A138 654 PRELIMINARY DESIGN FOR HARPOON SHIPBOARD COMMAND-LAUNCH CONTROL SET SIMULATION(U) NAVAL POSTGRADUATE SCHOOL MONTEREY CA J R ESCHLIMAN DEC 83 1/2

UNCLASSIFIED F/G 16/4.2 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AD A138654

(2)

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

PRELIMINARY DESIGN FOR HARPOON SHIPBOARD COMMAND-LAUNCH CONTROL SET SIMULATION

by

Judd R. Eschliman

December 1983

Thesis Advisor: Ronald W. Modes

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. A138654	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Preliminary Design for Harpoon Shipboard Command-Launch Control Set Simulation		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis December 1983
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Judd R. Eschliman		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93943		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93943		12. REPORT DATE December, 1983
		13. NUMBER OF PAGES 111
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Harpoon Weapons System, Software Engineering, Simulation

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

To realize the full performance capabilities of the Block 1C version of the surface launched HARPOON cruise missile, modifications have been directed on the HARPOON Ship Command-Launch Control Set (HSCLCS), AN/SWG-1(V). The purpose of this thesis is to begin the design of modules for a simulation model which meets the specification requirements of the HSCLCS. These specifications are derived from stated U.S. Navy requirements, (Continued)

ABSTRACT (Continued)

some additional features proposed by the author, and features proposed in a previous thesis. The simulation model can then be used for testing and evaluating the proposed modifications and for training purposes.

Approved for public release; distribution unlimited.

Preliminary Design for HARPOON Shipboard Command-Launch Control Set Simulation

by

Judd R. Eschliman
Lieutenant, United States Navy
B.S., University of Kansas, 1977

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
December 1983

Author: Judd R. Eschliman

Approved by: Ronald W. Modes
Thesis Advisor

Ronald B. Kurth
Second Reader

Chairman, Department of Computer Science

K.T. Marshall
Dean of Information and Policy Sciences

ABSTRACT

To realize the full performance capabilities of the Block 1C version of the surface-launched HARPOON cruise missile, modifications have been directed on the HARPOON Ship Command-Launch Control Set (HSCLCS), AN/SWG-1(V). The purpose of this thesis is to begin the design of modules for a simulation model which meets the specification requirements of the HSCLCS. These specifications are derived from stated U.S. Navy requirements, some additional features proposed by the author, and features proposed in a previous thesis. The simulation model can then be used for testing and evaluating the proposed modifications and for training purposes.

TABLE OF CONTENTS

LIST OF FIGURES

I. INTRODUCTION

With the introduction of additional performance capabilities in the Block 1C version of the HARPOON cruise missile, the present HARPOON Ship Command-Launch Control Set (HSCLCS) has proven to be inadequate for controlling this new HARPOON missile. Therefore, modifications have been directed by the Chief of Naval Operations, to take advantage of these supplementary features. The system specifications have been set forth by Naval Sea Systems Command.

Since the HSCLCS must be redesigned to facilitate the system specifications, it is readily apparent that a simulation model should be developed to test the system specifications and, once determined to be a usable model, to be further used for training purposes. The development of such a model allows an experimenter to play with the system, and investigate potential problem areas, as well as, encourage the process of innovation.

In designing a simulation model of a real system, the goal should be to conduct testing to understand the behavior of the system or to evaluate various strategies of system operation under consideration. A further goal should be for the model to be used for training purposes when it is not feasible or cost effective to use the real system. The art of modeling encompasses the ability to classify the problem, abstract the essential features, and then elaborate on these, to produce a model where useful approximation results. Special care must therefore be taken, to ensure that the model is an accurate and viable representation of the real system. To meet this end, certain criteria must be met in support of the development of a proper simulation model, including:

a. ease in understanding by the user.

b. a purpose or goal directed model.

c. ease of control and manipulation of the model.

d. model completeness on important issues.

e. no allowance for nonsense answers.

f. ease of model modification.

Keeping these criteria in mind, and realizing that simulation modeling is a learning process, the task of model development may begin.

II. HARPOON SHIPBOARD COMMAND-LAUNCH CONTROL SET (HSCLCS) SPECIFICATIONS

This chapter will summarize the system specifications as set forth in Reference 1, and as developed by Reference 2 and Reference 3. This phase will present the existing system, the needs of the existing system, and technical constraints imposed on hardware and software considerations.

A. PRESENT HARPOON WEAPON SYSTEM (HWS)

The HARPOON Weapon System (HWS) was developed to meet the needs of the Navy's anti-ship mission. This system is deployed on fast attack submarines, several type aircraft and surface combatants. The HWS's purpose is to provide all-weather, over the horizon, day/night anti-ship capability. It is composed of the missile subsystem, the associated launcher subsystem and the command and launch control subsystem. This thesis shall be primarily concerned with the latter.

The HARPOON missile utilizes an in flight low-level trajectory with a pop-up feature during it's terminal phase. It has an active radar seeker with counter-counter measure capability to assist in attacking over the horizon surface targets.

In the ship-launched version, the HARPOON missile utilizes either third party or onboard sensor data for targeting purposes. Then, since the missile requires no further information after launch, it is considered a "launch and forget" type weapon.

For the shipboard configuration, the HWS data processing and control functions are provided by the HSCLCS. The HSCLCS has three operating modes: casualty, normal, and training. In the normal mode, the HSCLCS provides the following major functions:

- Distribution of power to various HWS equipment.
- Selection and application of missile warmup power.
- The ability to conduct various automatic and manually initiated tests which confirm the operability of the HSCLCS.
- Selection, transfer, processing and display of target data.
- Coordination of the selection of tactical missile mode and type of fusing.
- Selection of the launcher cell containing the intended HARPOON missile.
- Initialization of the selected missile and the supervision of the exchange of data between the missile and other HWS equipment.
- Control of all missile firing activities.

These functions are carried out primarily by the HARPOON Control Console (HCC) and the Weapon Control Indicator Panel (WCIP).

The HCC encompasses most of the system-unique command and launch subsystem equipment. This includes the Data Processor Computer (DPC), a 16 bit microcomputer, and a Data Conversion Unit (DCU), an analog digital converter. These two components perform data conversion and processing, and provide an interface with current ship sensors or external equipment.

The WCIP provides the operator with visual status information of the fire control solution. It further provides the operator with manual input capability.

The DFU executes an assembly language program to provide the following services:

- Validation of launch envelope parameters.
- Missile command generation for implementation of missile control parameters including ship's attitude, search pattern orders, engine starting, flight termination range, altimeter setting, and various selectable flight trajectory and maneuvering modes.
- Missile testing prior to launch.
- Pre-launch sequencing and timing.
- Data formatting and transfer synchronization.

The DCU processes all digital and analog signal conversion. It further provides interfacing of target data inputs for the Naval Tactical Data System (NTDS) and it also provides for ship motion parameter conversion.

B. CURRENT DEFICIENCIES IN THE HSCLCS

With the introduction of new enhancements in the HARPCON missile, new command and control problems have also been introduced. The current WCIP cannot accommodate these new capabilities. Further, the operator is not provided with sufficient facilities to direct and execute a well-planned HARPOON attack. These new capabilities mandate a substantial change in the hardware and/or software of the existing HSCLCS. Since current software is written in machine language, it is extremely machine dependent. This, coupled with the added difficulty that different hardware configurations exist for subsurface, surface and air launches, further compounds the problem of software standardization. Reference 2 and Reference 3 set forth existing deficiencies in the current HSCLCS. These include:

- Full tactical control of missile variants (the pre-launch selections) are not available to the existing WCIP.

- The WCIP provides inadequate control for a well coordinated, multi-ship or multi-platform attack against a single surface target.
- The WCIP provides inadequate control for a multi-missile (SALVO) attack against a single surface target.
- The WCIP does not incorporate existing intelligence information (e.g., target class, course, speed, sector of vulnerability) into the engagement planning process.
- No computer-aided engaged planning is implemented.

For engagement planning, the HSCLCS has the following deficiencies:

- Insufficient information is displayed at the WCIP to permit the operator to evaluate the quality of an engagement plan (e.g., probability of acquisition).
- Insufficient information is displayed at the WCIP to provide accurate data, implying risk to unintended targets during booster drop, flyout and target acquisition.
- The WCIP provides no display of planned trajectory, flight path or seeker search patterns.
- The HSCLCS does not provide computer-aided engagement plan quality and safety analysis.
- The WCIP provides no status information on available missiles and associated launcher.
- Only track data for one track can be stored, with no capability for multi-track retention.
- No means are currently available to provide corrections essential to missile performance for the environmental parameters such as wind, rain and sea state.

With these deficiences, the training mode is, at best, minimal. Since there is a general lack of realism, especially in the graphical representation of the engagement picture, the operator has little ability or inclination to improve his proficiency.

C. HWS CONSTRAINTS

Modifications to the HSCLCS should take advantage of, but not necessarily be limited to, the following new missile capabilities:

- Waypoint selection.
- Range and bearing (RBL) search pattern expansion direction selection.
- Terminal attack mode selection.
- Maximum range increase.
- High-altitude flyout
- Pre-search sea skim.

With these capabilities in mind, Reference 1 has set forth the modification and specification limitations of each of the components in the HSCLCS.

1. <u>HARPOON Control Console</u> (<u>HCC</u>)

The HCC may be modified as required to accommodate power and digital interfaces to the WCIP, and to provide integral mounting with the WCIP. In addition, the HCC must meet the specification requirements of Appendix C.

2. <u>Data Conversion Unit</u> (<u>DCU</u>)

The DCU modifications are limited to the removal of circuit cards in order to provide required functions for the WCIP. It will provide an interface with the ship's motion systems, target detection systems and missile launch equipment.

3. <u>Data Processor Computer</u> (<u>DPC</u>)

The DPC is a general purpose computer with ultraviolet erasable programmable read-only memory (UV-EPROM). This computer provides weapon control solutions to the missile and provides direct real time control of the WCIP.

It also conducts interlock computations to prevent launch when the ship roll and pitch are excessive, launcher pointing and stabilization orders and conducts self-testing on the WCIP or HARPOON missile when directed. The modified DPC shall consist of (a) a new Central Processing Unit (CPU), which will provide the computational speed required of the WCIP engagement planning functions, (b) a minimum of 110,000 sixteen-bit words of UV-EPROM, 16,000 sixteen-bit words of random access memory (RAM), and 2,000 sixteen-bit electronically erasable programmable read-only memory, and (c) an RS 232 serial interface for use with external devices providing training or data extract functions. The DPC may be further modified to support the engagement planning functions as specified in Appendix C.

4. <u>Weapons Control-Indicator Panel (WCIP) Graphic Display System</u>

The WCIP Graphics Display System (GDS) shall consist of a plasma graphic display with embedded microprocessor and twenty manual entry buttons under software control. This function will allow the operator to plan, evaluate and execute a HARPOON engagement and to conduct training. The display is to provide the operator with a visual depiction of the tactical engagement situation and concurrent readouts of necessary engagement data. It is noted that a prototype display has already been developed, and is currently under evaluation.

5. <u>Weapon Control-Indicator Panel</u>

The WCIP will be required to provide all the functions of the specifications listed in Appendix C. It must provide operator/HSCLCS interactive functions required for engagement planning. It will provide visual status information on the HARPOON missile and launcher, and the manual

controls for selecting missile initialization functions/ parameters and firing of all *HARPOON* missile variants. Although target data entry for missile initialization is normally automatic through the DCU, alternate manual entry must be provided for from the WCIP.

D. PROPOSED SOFTWARE PLAN

Reference 2 and Reference 3 set forth consecutive refinements to a software plan. A requirements analysis was conducted to provide a foundation for the flow and structure of information and to identify interface details within existing design constraints. To accomplish this analysis the use of a data flow diagram (DFD) was used. This type of diagram is a graphical aid for demonstrating data flow during the designing of a software system. A general outline of a DFD consists of the following:

1. <u>DFD Attributes</u>
   - Information flow can be represented by a *DFD* for any system.
   - Each transformation in a DFD may require refinement to realize a complete understanding of the transformation.
   - Data flow must be emphasized without regard to control of data.
2. <u>DFD Symbols</u>
   - Information flow is identified by a labeled line from the source to the sink with an arrowhead pointing in the direction of flow.
   - Data transformation is represented by a circle.
   - Information sources and sinks are displayed as rectangles.
   - Stored information (i.e., data bases and files) are represented by two parallel lines.

3. DFD Usage Guidelines

- The first layer of the DFD is the system module.
- The second layer of the DFD is a generalized overview of the DFD.
- All items in the DFD including arrows must be labeled.
- Information continuity is required for all DFD refinements.
- Refine only one item at a time.
- When uncertainty exists as to whether further refinement, assume that the possibility exists.
- Follow data flow from left to right.
- A transformation may output control data for a subordinate module. This control data does not represent control structure and therefore is not control flow.

Figures 2.1 through 2.4 represent a refined development of the HSCLCS by the data flow method.

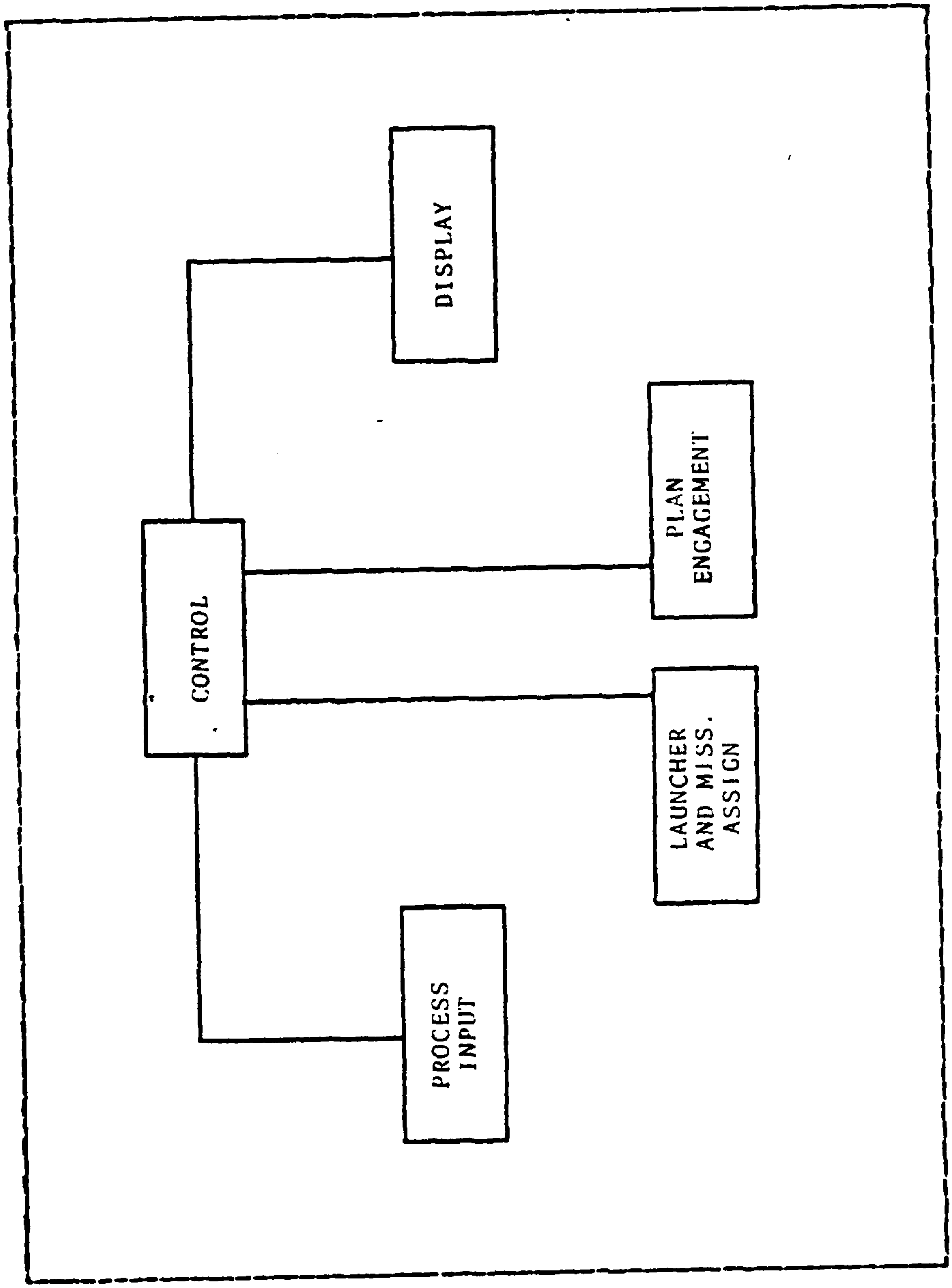


Figure 2.1 Top Level Module Control.

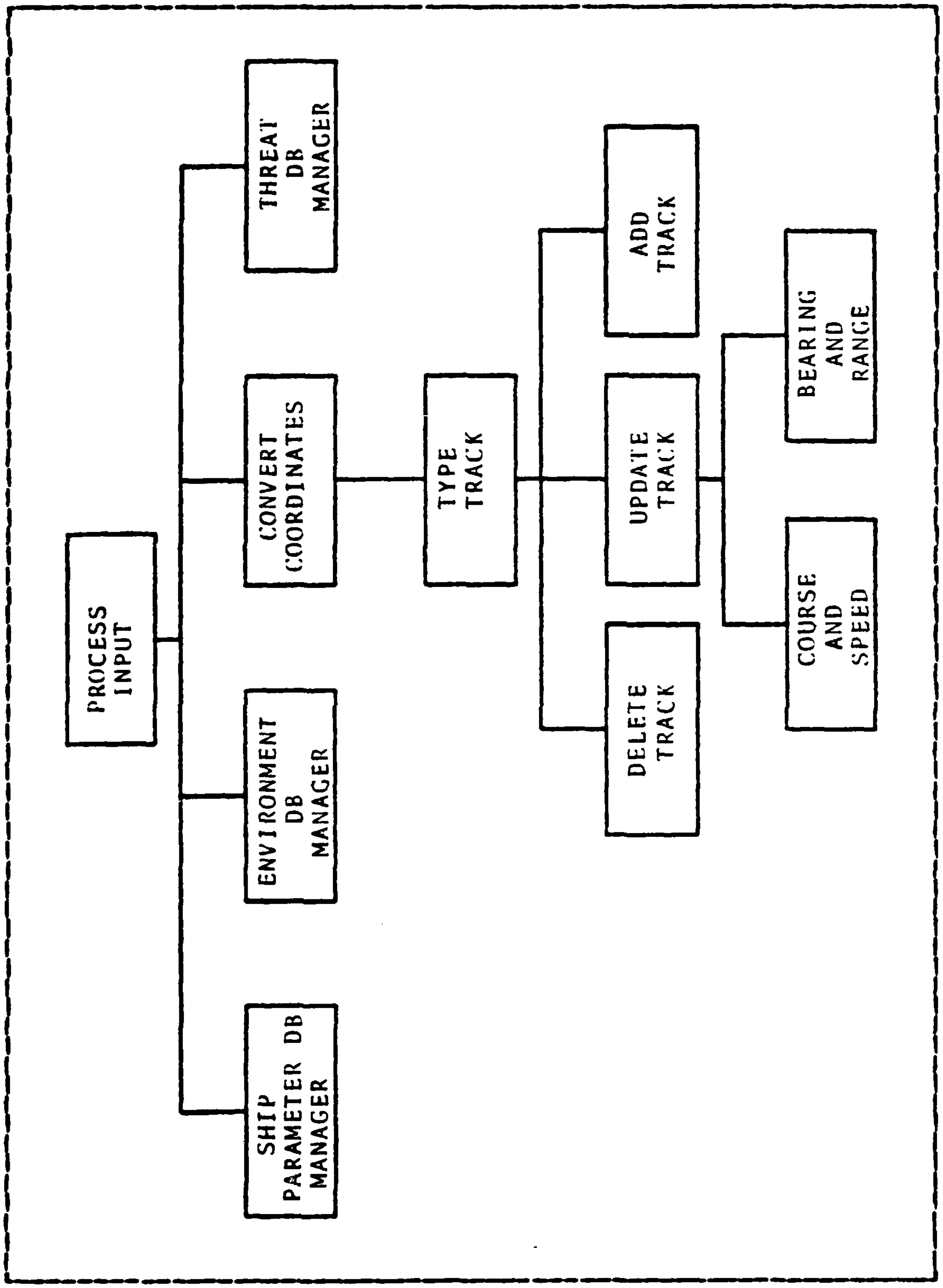


Figure 2.2 PROCESS INPUT.

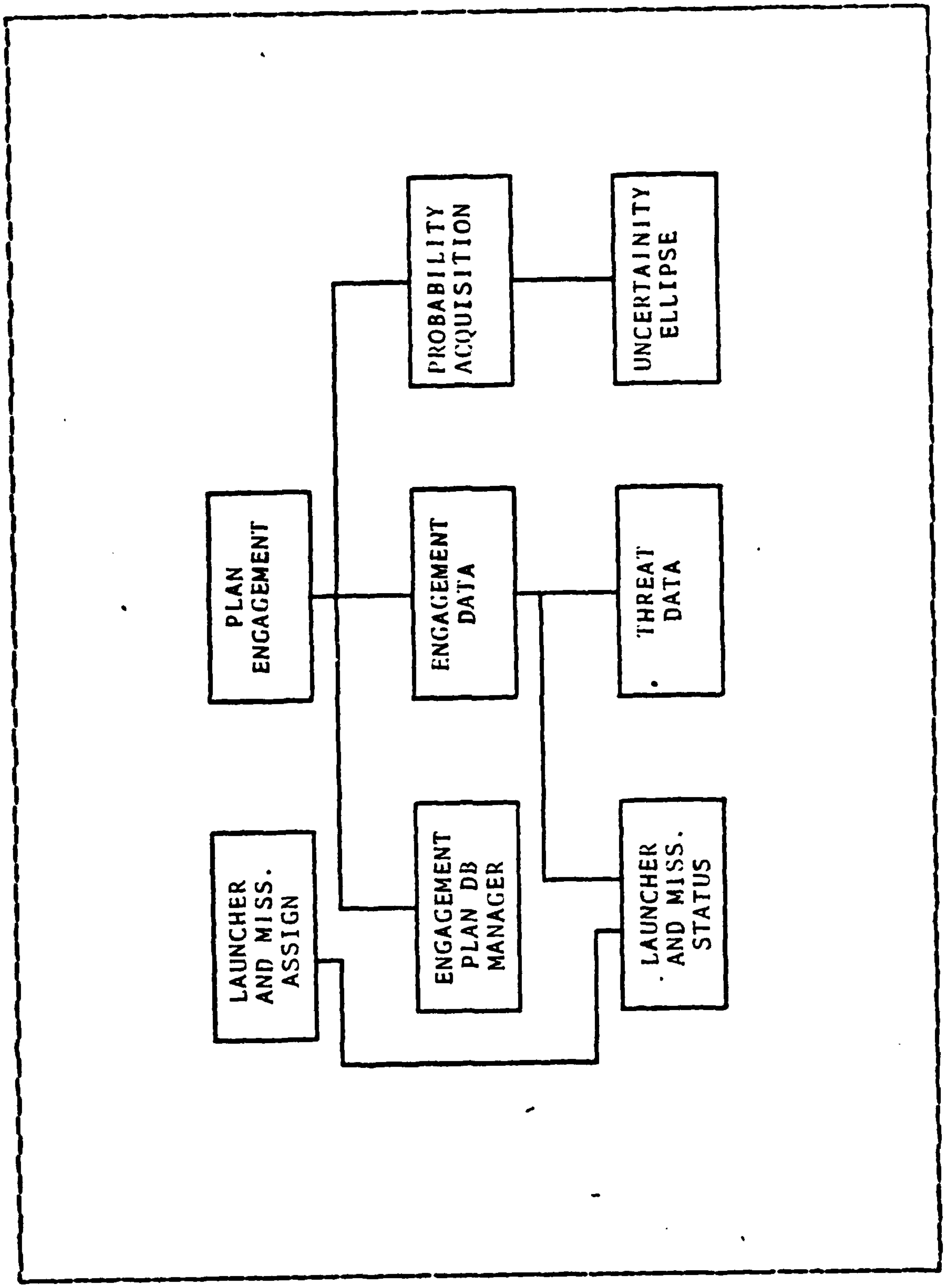


Figure 2.3 PROCESS ENGAGEMENT.

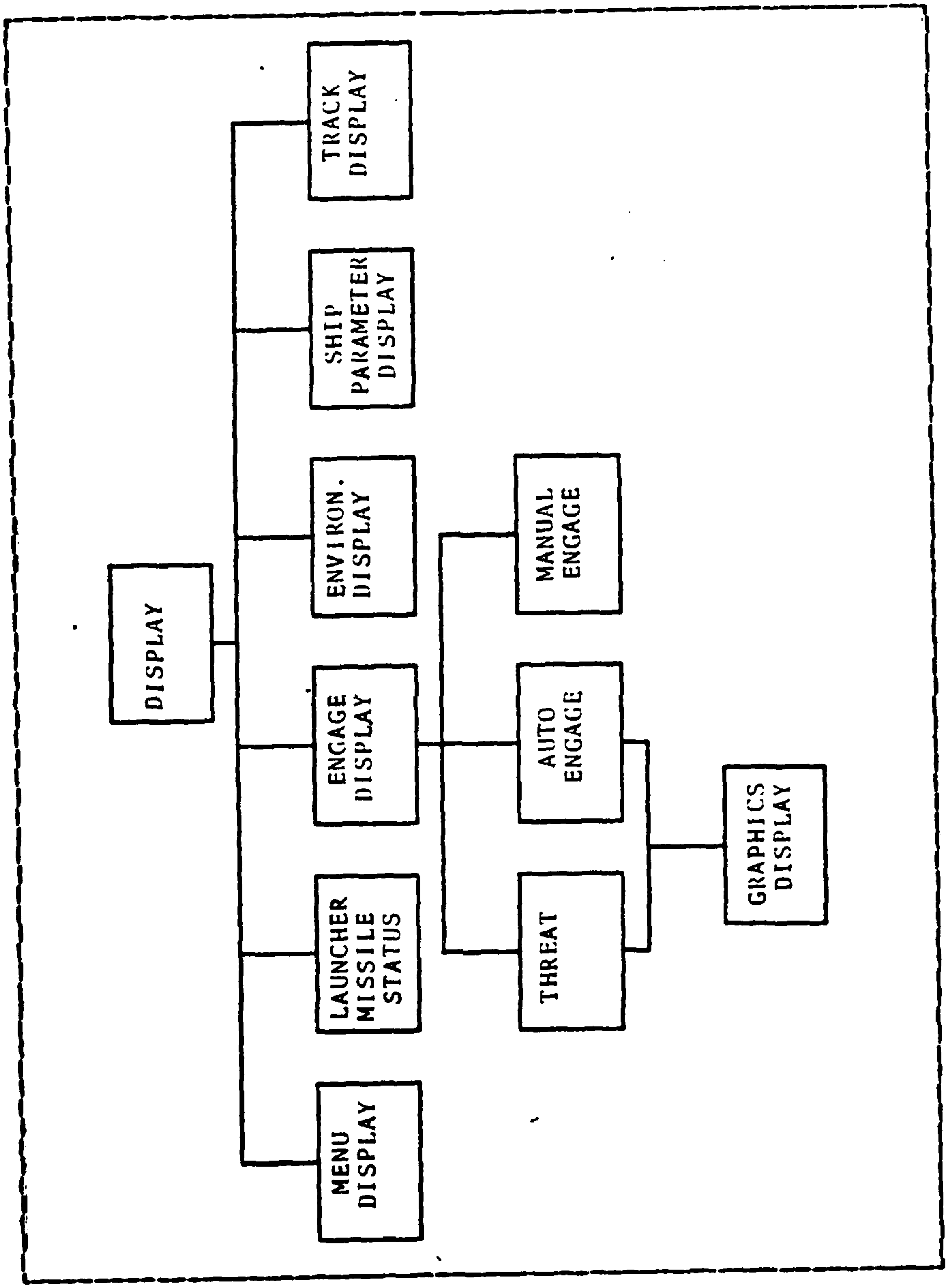


Figure 2.4 PROCESS DISPLAY.

III. MODELING PROCESS

In creating a simulation model of the HSCLCS, the assumption is made that the simulation will be used to examine the properties of the HSCLCS, and, if found dependable, used as a training device. In developing this model, a number of intermediate steps may be identified to assist in the model development. Several of these steps will be developed here, but others must wait until actual model implementation conducted under other theses.

A. SYSTEM DEFINITION

At this step of the modeling process, a determination must be made of the boundaries (i.e., what will be simulated) utilized in developing the system. Since a formulation of the simulation model may change as it is being developed, the conditions set forth at this step cannot be considered solid. As new information becomes available, these conditions must be amenable to change.

1. Boundaries

The HSCLCS is a major component of the HARPOON Weapon System (HWS) comprised of two elements: the HARPOON Control Console (HCC), and the Weapon Control - Indicator Panel (WCIP). It's primary purpose is to provide the capability to initialize and launch existing HARPOON missiles. In it's present configuration, the HSCLCS provides a weapon control solution, missile initialization, launcher control and missile firing functions. With new modifications, the HSCLCS should also provide engagement planning and data/training extract capability.

The HCC performs most of the data processing and conversion necessary for missile launch. The WCIP provides visual information to the operator during the fire control solution formulation, and further provides for manual control by the operator. In the simulation model development, it would appear that the HSCLCS would be the sole object of simulation. However, since the HSCLCS obtains information/data from other ship sensors or from proposed or current manual input, the model must allow for input of simulated ship sensor information/data either manually or automatically.

2. Model Formulation

The next task is that of model formulation: reduction of the real system to some logical flow pattern. The model formulation should neither oversimplify the system nor overspecify it so as to appear awkward or become extremely expensive. The latter case is usually the problem area. In this simulation, the model may be necessarily bulky due to the need to simulate the entire HSCLCS subsystem.

In setting forth the model formulation, certain criteria must be incorporated to ensure a good simulation. Because the final result of this simulation is to be more along the line of a prototype development that as an analysis tool, the initial structure for the model formulation as set forth in the data flow diagrams of Chapter 2 can be used. It would then appear that each of the boxes in the data flow diagrams can be developed into a module for the simulation. Several criteria are met for a good simulation by proceeding in this manner.

The first criteria is that the model becomes easily purpose or goal directed. Each module will be developed to carry out a specific task. If the desired goal of the model changes, such as for training, then a minimum number of

changes will be required. This coincides with the important criteria of model modification. Through the creation of modules from the data flow diagrams, using the transaction transform analysis methodology described in Reference 4, the effects of any changes to either the design specifications or to the system design will be minimized.

Modularization can also lead to ease of control and manipulation of the model. If the modules are developed correctly, that is, through proper abstraction and good interfaces, then the designer will only need to know what the module does and not necessarily how it does it. This in turn leads to a better understanding of the system by reducing the complexity at each level of the model. This will further allow ease in determining completeness on all important areas of the model.

Other criteria for a good simulation must be completed at the implementation stage of model development. These include ease in understanding by the user and no allowance for nonsense answers. The former will require carefully designed interaction with the user, whereas the latter will require input parameter checking.

In a general overview, the model can probably best be split into 4 modules: a) data processing unit (DPU), b) data conversion unit (DCU), c) weapons control-indicator (WCIP) panel graphics processor and d) weapon control-indicator panel controls. The main desire is to check the use of the WCIP graphics processor and controls, however, since much of the input is through the DPC and DCU, they too will have to be simulated. The DPU and DCU may carry the added burden of simulating the ship sensors, or may solely provide data in a refined form. The graphics processor will probably be the most difficult to simulate, and may not in fact be completely necessary. However, to ensure some sense of reality, the graphics module should closely model an actual display.

Since a first design refinement of the system has already been made, Ref. 3 there appears to be no real need to make major changes at this point. Now the task of defining the structures for the data bases identified in Appendix D, and the modules identified in Appendix E, can be made.

3. Databases

All of the data bases in Appendix D appear to be sufficient for the simulation. This assumption does not, however, prevent changes in later stages of the model development. If an element in the data base requires a default or initial value, it will be accomplished by one of the modules listed in Appendix E.

a. Environmental Data Base

The Environmental Data Base is used to store information about current weather conditions, to be used in obtaining a practical engagement solution. This data base will always reside in main memory. It will consist of a single record with eight fields:

Visibility	:	integer range 0..30(miles
Sea State	:	integer range 0..10
Wind Direction	:	integer range 0..359 (degrees
Wind Speed	:	integer range 0..100 (knots
Relative Humidity	:	integer range 0..100 (percent
Temperature	:	integer range -100..150 (degrees F
Barometric Pressure	:	integer range 900..1100(millibars
Precipitation	:	string(yes,no

The following default values will be made automatically prior to any manual entry from the operator, or automatic entry from selected ship sensors.

Visibility	:	1
Sea State	:	1
Wind Direction	:	000

Wind Speed	:	0
Temperature	:	50
Barometric	:	1020
Precipitation	:	No

After initialization, the operator may input the current environmental conditions manually or allow automatic updates from own ship sensors.

b. Launcher and Missile Status Data Base

This data base will consist of an array of records equal to the number of cannisters available for launching missiles. It will always reside in main memory. Each record will consist of the following four fields.

Launcher Number	:	integer range 1..#launchers
Cannister Number	:	integer range 1..8
Missile Type	:	integer range 1..7
Launch Inhibit	:	integer range 0..1

The following values will be assigned upon activation of the system. The values will be selected from a file in secondary storage. The file in secondary storage may be modified to accommodate any type of platform and any type of missile loadout. When any changes are made by the operator to the data base, they may only be accomplished using a special code.

Launcher Number	:	current launcher number
Cannister Number	:	current cannister number
Missile Type	:	type missile in current cannister (a value of 7 indicates the launcher is empty and requires launch to be inhibited for that cannister)
Launch Inhibit	:	0 (inhibited)

The size of the structure holding this data base in main memory must be variable to allow for the requirements of different platforms. One method to permit this would be to have the first item read from the secondary storage file indicate the size of the array (i.e., the number of available cannisters). If for some reason the size needs to be changed, such as for an inoperable launcher, the operator will have the capability of changing the size.

c. Menu/State Data Base

This data base will reside permanently in main memory. Immediately upon power up, this data base will be initialized. The Control Module will make a call to the Menu/State Display Module, notifying it that the current state, or process instance, is 0. This will cause the Menu/State Display Module to initialize the data base and enter state 1. The data base will consist of an array of records. Several of the fields will be of variable length. The fields will be arranged in the following manner:

```
State              :  integer range 1..maxstate
Number Options     :  integer 1..maxoptions
Menu Display       :  array (2,Number Options)
                          Option
                          Display Location
```

After the data base has been initialized, the first menu that will be displayed will be the main menu. Thereafter, whenever a menu selection is made, the state will be identified and sent to the Menu/State Display Module.

d. Ship Parameter Data Base

The Ship Parameter Data Base will consist of information about the operators own ship. It will contain a single record with four elements, and will reside in main memory.

Course	:	integer range 0..359
Speed	:	integer range 0..maxspeed (maxspeed is maximum speed of the platform which must be tailored for each individual platform).
Latitude	:	record degrees : integer range -90..90 minutes : integer range 0..60 seconds : integer range 0..60
Longitude	:	record degrees : integer range -180..180 minutes : integer range 0..60 seconds : integer range 0..60

All elements will be initialized to 0 after which the operator may input changes manually. Automatic changes may also be made via ship sensors (e.g., pit sword, dummy log, gyro, navigation equipment).

e. Threat Data Base

This data will always reside in some secondary storage since it will only be needed when requested. The potential for this data base becoming very large is readily apparent. This data base may only be changed with special permission. For the purpose of this simulation it will consist of a file of records with six elements.

Ship Name	:	string
Nationality	:	string
Ship Class	:	string

Weapons	:	string
ECM Equipment	:	string
Engagement Method	:	string

All information in this data base may be changed by the operator using a special code. The information in this file may be of classified origin. The data base should have the capability of being accessed for specific information by the Ship Name. Further, to facilitate flexibility, a list of ships should be capable of being accessed by using the Nationality or Ship Class elements.

f. Engagement Data Base

This data base will be utilized in conjunction with the Track Data Base. Since there will be a need for information in both data bases to be identified by a track number, it appears appropriate to store both data bases as one. However, each data base's manager will only have access to a portion of the information. To allow for engagement of up to 20 tracks, an array of 20 elements will be needed. Each of the array elements will consist of a record with sixteen fields.

Track Number	:	integer range 100..3199
Type Track	:	integer range 1..7
		(1=Surface Friendly)
		(2=Air Friendly)
		(3=Subsurface Friendly)
		(4=Surface Hostile)
		(5=Air Hostile)
		(6=Subsurface Hostile)
		(7=Unknown)
Type Engagement	:	integer range 0..1
		(0=Manual,1=Auto)
Class of Vessel	:	integer 0..1
		(0=Large, 1=Small)
Bearing	:	integer range 0.359(degrees

Range	:	integer range 0..maxrange(miles
Latitude	:	record
		Degrees : integer range -90..90
		Minutes : integer range 0..60
		Seconds : integer range 0..60
Longitude	:	record
		Degrees : integer range -180..180
		Minutes : integer range 0..60
		Seconds : integer range 0..60
Course	:	integer range 0..359(degrees
Speed	:	integer range 0..max(knots
Number Miss Fire	:	integer range 0..8
Firing Sequence	:	array range 0..max
Type Missile	:	integer range 1..4
Flight Path	:	TBD
Waypoints	:	TBD
Missile Selected	:	integer range 1..max

4. Level 1 Modules

Level 1 is the highest level in the simulation. It corresponds to the Control Module of Figure 2.1. It is this module which will be responsible for controlling all lower level modules and ensuring that the simulation always remains in a consistent state. Immediately upon activation of the system, the Control Module will be responsible for initializing all of the data bases with their required default values. This may require the addition of a module to perform initialization and to input default values as the operator makes additions, deletions and changes to appropriate data bases.

The Control Module will also be responsible for ensuring that sufficient parameters are available when calling lower level procedures. It will be at this level that lower level modules requiring a "time" field for their data structures will obtain the correct time. This need for a "time" field will be addressed in the next chapter.

The simulation must further meet the specifications of Appendix 1, requiring a training mode and data extract capability. These features should be incorporated to allow the user to specify a training or a full simulation mode. If the training mode is selected then a file in secondary storage will be opened. Whenever any change is made in any data base, the change will be entered sequentially, with the time of the change in the secondary storage file. This will allow for later retrieval of the data so the operator may study and evaluate his training session.

5. Level 2 Modules

There are two modules at this level: the Process Module and the Display Module. Normally the Launcher/Missile Assignment Module would be at this level, however, it has been moved to level 3 to allow for manual input of launcher and missile assignment. This will permit the operator to fire a missile quickly should the tactical situation dictate, without having the engagement analyzed. This does not prevent automatic engagement which may also be dependent upon time constraints and/or the tactical situation.

a. Process Module

The Process Module will control the selection of lower level modules for addition, deletion and making changes in appropriate data base records. The only information which the Process Module should return to the Control

Module is information concerning errors in improper parameter passing and records not found error messages. If the operator selected the training mode at the Control Module level, the Process Module should have been passed this information just one time. This will allow all changes, additions and deletions to be written to a secondary storage file. Further, if the information is passed just once, then valuable time will not be wasted at each call to the Process Module to see which simulation mode has been selected. The Process Module should also prevent unauthorized changes in the Threat Data Base Manager Module. The latter requirement may not be needed, as will be addressed later. The Process Module will carry the added burden of simulating ship sensors, NTDS and third party information. These tasks should, if possible, be conducted concurrently.

b. Display Module

Here is the module that has the potential to become very large. The Display module must display all requested information without destroying any data in any of the data bases. This module should only pass information to the control module concerning errors in requests for displays that do not exist. This module will undoubtedly require additional refinement when the model progresses to the model enters the testing phase of the model simulation.

6. Level 3 Modules

Most of the data base managers occur at this level, with the exception of the Plan Engagement Data Base Manager. The majority of the graphics display modules are also at this level. It is here that extreme care must be made in granting write access to the data bases and ensuring that changes to fields in the data bases are within appropriate ranges. This might be accomplished as an input control.

a. Ship Parameter Data Base Manager

This is the only module which will have write access to the Ship Parameter Data Base. After initialization of the data base, the operator will be able to change the fields manually. Automatic updates may be made through the Control Module from own ship sensors (e.g., gyro, navigation equipment). The Control Module will ensure that updated information is passed to this module as required, and upon every speed or course change. During the period between updates ordered by the Control Module, the Ship Parameter Data Base Manager will cause updates every minute based upon current course, speed and position in the data base. This will permit those modules with read access to the data base to have reasonable confidence in the accuracy of the data. If the Control Module was unable to obtain an update from own ship sensors, the operator will be notified that the data may be out of time tolerance. This might be accomplished by prompting the operator for input at the required time, and notifying him what equipment has failed.

b. Environmental Data Base Manager

This module is responsible for updating the Environmental Data Base. It is the only module with write capability to this data base. Since most of the data in the Environmental Data Base will remain accurate for several hours, with the exception of wind speed and direction, there is no mandatory updates unless specified by the operator. In the case of mandatory updates, the operator may specify time intervals in the Control Module, to permit checking of ship sensors for wind speed and direction only.

c. Convert Coordinates Module

This module will provide a capability for the Track Data Base to be updated either manually, from own ship sensors, or via the NTDS link. As this module receives information, it will determine the coordinate system represented by the data. The Convert Coordinates Module will then change, if necessary, this data to coordinates in the reference system the operator has selected (i.e., true, NTDS grid, geographic). If this module has been accessed solely to add or delete information, then no conversion should take place. At any point, if the operator chooses to change the reference system, this module will make the appropriate conversion of data for the display.

d. Threat Data Base Manager

The Threat Data Base Manager will be used to change data in the Threat Data Base. It is the only module that will have write access to this data base. If a change is made to this data base, it should be done prior to commencing training or at times when the system is solely activated for this purpose. The major reason for this is the potentially large size requires that it be stored in secondary storage, thus significant time will be required to access the desired data to make changes. For the purposes of this simulation, this model can most probably be deleted. The reason for this is that since only twenty keys are required to be under software control, the real system will probably be updated off line. This should not detract from the overall testing of the system in later phases. It is anticipated that secondary storage will be either a hard disc or bubble memory.

e. Menu Display Module

This module will select the appropriate menu from the Menu/State Data Base and display it on the screen. The Menu Display Module will always keep track of the current state of the system, and only display the available alternatives. No alternatives should be made available that can not be carried out (e.g., allowing a change to the Environmental Data Base when the Ship Parameter Data Base is currently accessed). All menus should have the capability of escaping to the Main menu. This feature will always allow the operator to escape in order to fire a missile. Since this allows the operator to escape before all necessary data might be input, there will be times when changes to data bases should not be made until all the required information has been entered. An example is adding a track and assigning a missile without providing some information concerning range and/or bearing. In this case, no track should be added until minimal information has been provided. The only information that must be passed to this module is the current state. Since the specifications only call for 20 keys to be under software control, many of the keys will be required to have multiple menu assignments. The key will therefore have to be displayed with the menu selection to which it pertains.

f. Launcher Missile Status Display Module

Here the operator is provided with the ability to check the current missile/launcher status for the entire system, or any portion thereof. The Launcher Missile Status Data Base may be read by this module, but may not make any changes to it. If the operator desires, he may select display for a particular missile, all missiles in a specific launcher, or for the status of all missiles.

g. Environmental Display Module

When called by the Display Module, the current values for the environmental conditions from the Environmental Data Base will be displayed. This module will only have read capabilities on the data base. However, the operator will be able to select from one of the menus to make a change.

h. Engagement Display Module

The Engagement Display Module controls several lower level modules. It's main purpose is to decide which module to select based upon the operators desires. It will cause information from the Threat Data Base to be displayed, or information about the current engagement plan. If the operator selected the manual engagement display, he will have the opportunity to input his own engagement plan and see what will happen, prior to any changes being entered in the data base. This will present the operator an opportunity to check his plan before instituting it.

i. Ship Parameter Display Module

This module will display all information currently in the Ship Parameter Data Base. It will only be able to read from the data base, and present the information in an intelligible form for the operator to understand. A menu selection will be available when this module is called, to permit the operator to institute a change to the data base.

j. Track Display Module

This is the last module at level 3. It's purpose is to display all tracks currently in the Track Data Base. Any time any change is made in the Track Data Base,

the display should reflect the change. This information should be displayed at all times for the operator. To facilitate this, all menu displays and other information must be displayed on areas of the screen outside the track display. The one exception to this is the engagement display which must be superimposed on the track display.

7. Level 4 Modules

This level of modules controls the type of track update, manual missile assignment, automatic and manual engagement planning, and the display of the engagement plan. These modules will be used almost extensively for selecting lower level modules to process data.

a. Type Track Module

Little is required of this module. It must decide if a track is being accessed for addition, deletion or update, and then select the proper lower level module. One task that might be useful to incorporate at this level, is to maintain a count of the number of active tracks in the Engagement Data Base. By keeping a count at this level, unnecessary searching of the data base can be prevented when a track addition is needed and the data base is full.

b. Launcher Missile Assignment Module

This module is one of the most significant. It's purpose is to allow the operator to bypass engagement planning modules and quickly select and launch the desired missiles. The operator will have the option of assigning any missile to a specific track. After launch the Engagement Data Base must be updated. This may be done by updating the Launcher Missile Status Data Base with information that the cannister is now empty after the missile has been fired. A drawback to this is that by allowing the

operator to fire in this manner, he will not be able to obtain an engagement display for this firing. The operator may select this module only from the main menu.

c. Plan Engagement Module

This module automatically generates the optimum type engagement for all designated tracks unless manual input has been specified. The Launcher and Missile Assignment Module may also bypass this module whenever rapid firing of a missile is mandated. Information will be obtained from the Engagement Data Module and Probability of Acquisition Module, and a solution identified. The solution will then be entered in the Engagement Plan Data Base. This information may be displayed by the operator, whenever he desires.

d. Threat Display Module

Upon entry into this module, the operator will be prompted for the method by which desired information should be obtained from the Threat Data Base (i.e., via ship name, nationality, or ship class). After the operator has made his selection, the module will access the Threat Data Base in secondary storage, and display the requested information. The operator will have the opportunity of requesting additional information if he originally requested information by nationality or ship class. If the information requested will not fit on the screen, the operator should be able to conduct paging until he obtains the necessary information. He must be allowed the opportunity to escape at any time. The menu selection should therefore allow for paging until all the information has been displayed, or the operator has requested an escape.

e. Automatic Engagement Display Module

Here, all information concerning the planned engagement for a desired track is displayed. This will allow the operator to see before hand if the type engagement is proper, and if not, make the appropriate adjustments. Information obtained from the Engagement Plan Data Base will be passed to a Graphics Display module so that the actual display may be made. The reason for passing the information, instead of displaying it immediately is that the Manual Engagement Display Module will use the same graphics module. The will help prevent a duplication of code.

8. <u>Level 5 Modules</u>

This level might be considered the work level. It is here that the majority of data is added, deleted or changed. Information is also obtained at this level for forming an automatic engagement solution. The Graphics Display Module also does all of the work at this level to present the operator with an engagement display.

a. Delete Module

For ease in deleting a track, the delete module will locate the desired track in the Track Data Base and set the Track Number and Track Type fields to zero. This will cause the record to become free for future use. If the track is not located, the operator must be notified, in the event that an improper entry was made. There is no need to zero all the fields in the data base, since other modules with read access will know by the Type Track field, that the record is inactive.

b. Update Track Module

Here the operator must be given a menu selection to permit this module to decide if he wants to update course and speed and/or range and/or bearing. He should have the capability of update any or all. He must then identify the track he desires to update. If the update is automatic, the Update Track Module will decide which lower level module to call.

c. Add Track Module

If the Add Track Module is called, it will search the Track Data Base for the an available Track Number field to have all zeros. Zeros in the Track Number field indicated the record is free for use. All information passed will this be placed in this record. If no record is available, the operator will be notified, that the data base is full. An additional requirement for the track data base, may be the need to place a priority on each of the tracks. This would allow the removal of the lowest priority track if the data base is full and a higher priority track is designated.

d. Engagement Plan Data Base Manager Module

The only purpose for this module is to enter information into the Engagement Plan Data Base as specified by the Plan Engagement Module. It is the only module to have write access to the Engagement Plan Data Base. If the operator had decided on a manual type engagement of his own while in the Manual Engagement Module, this information would have been passed up and down through the Control Module.

e. Probability of Acquisition Module

This module will generate the probability of acquisition by obtaining information from the Uncertainity Ellipse Module. Information that must be passed to this module is type missile, search pattern, type target, and target ECM capabilities.

f. Graphics Display Module

This module will probably be the most difficult and cumbersome module to develop. It must be set up to display all the information concerning engagement of all targets. It must not erase the information displayed by the Track Display Module, and it must be flexible enough to handle use from the Automatic Engagement Display Module and the Manual Engagement Display Module.

9. <u>Level 6 Modules</u>

This is the lowest of the module levels. At this level minor calculations are made, data is obtained from several data bases, or changes are made in data bases.

a. Course/Speed Update Module

If called by the Update Track Module, the Track Data Base will be accessed for a course and/or speed change. This module must ensure that all entries to course and/or speed are within proper range. If not the operator must be notified to reenter, or verify the the information to be correct. If the change was requested by an automatic update, and was out of range, the information must be displayed to the operator, so that corrections may be made or the request canceled.

b. Bearing/Range Update Module

This module will operate in the same manner as the Update Track Module, except it will only deal with the bearing and/or range parameters.

c. Launcher and Missile Status Module

The Launcher Assignment Module may obtain information directly from this module in order to select an available missile for immediate firing. This module will access the Launcher Missile Data Base and permit the operator to select any missile not inhibited for launch. Once the missile is fired an update must be made through the Plan Engagement Module in the event that the missile has been selected for another target in the Engagement Plan Data Base.

d. Threat Data Module

The only function of this module is to read data from the Threat Data Base, and provide the information to the Engagement Data Module. The module does not have write capability for the Engagement Data Base.

e. Uncertainity Ellipse Module

By taking parameters such as number of missiles to be fired, probability of acquisition of the target, and lethal capability of the missile(s) being fired, this module will return ellipses to determine the uncertainity of locating the target within each ellipse. This information will be provided up the line to the Plan Engagement Module for evaluation.

B. SIMULATION OVERVIEW

Since the objectives of each module in the simulation has been established, an overview of how the simulation should interact with the user is in order. This overview is not all encompassing, and should not restrict implementation on a broader or more conservative scale. Specifications set forth in Appendix C should, however, be maintained as closely as possible.

Immediately upon activation of the simulation, all data bases will be initialized to their default values. The user should not be required to provide any input to effect this initialization, with the possible exception of specifying the type platform (e.g., destroyer, frigate, cruiser). The exception could be randomly generated to keep the user from always using the same type platform.

After initialization has been completed, the user will be prompted for his desired mode of simulation: full simulation mode or training mode. The only difference in the selections is that the latter causes all changes to any data base to be entered in a secondary storage file for later purusal. The user will then be prompted for the current time in hours, minutes and seconds. This information will be used to start a real time clock. The clock will time the simulation, and provide random variants for the generation of ship sensor, automatic input and required manual input data. This latter feature will provide some sense of realism by requiring some information to be manually input by the user. Further, the use of theoretical frequency or a probability distribution is considered to be more efficient use of computer time and memory storage requirements than a canned table look-up procedure.

Throughout the simulation, the user will be randomly provided with information concerning all of the data bases. He should have the option to decide at any time to input additional information, provided the input makes sense. Additionally, provisions should be provided to simulate the failure of some of the ship sensors, but not sufficient enough to stop the simulation. An example is simulating NTDS link failure, but then providing data for manual input. One final interactive feature that should be provided, is not allowing automatic update to the information the user is manually updating, as opposed to information simply being accessed. The simulation will terminate whenever all missiles have been fired, or only unlaunchable missiles remain. The user may also terminate the simulation at any time.

One additional feature that might be advisable to add is a "freeze problem" feature. This would allow the simulation to be stopped during the training mode, to allow the operator to evaluate the problem. The operator could then resume the simulation at the same point that it was stopped.

IV. SYSTEM DESIGN LANGUAGE SELECTION

An important step in the model simulation is the selection of a system design language. It's purpose should be to show what program units are needed, and what interfaces each unit requires. However, by selecting a system design language, no restriction is placed on selecting an alternative programming language for actual implementation.

A. SELECTION GOALS

In selecting a system design language, no assumptions have been made about the type of computer that will be selected for implementation. For the purpose of this thesis, the selection of a system design language will be based upon several necessary qualities for practical software engineering. The primary goal of the design is that the solution meet the stated specifications. This goal is rarely met if the following software design qualities are not achieved; modifiability, efficiancy, reliability and understandability. The achievement of these qualities will enhance the attainment of a good simulation model as specified in Chapter 1.

1. Modifiability

Although difficult to realize, the ability to easily modify the the software is vitally important, because at some point a change in the specifications will become necessary. To effectively change a system, the current design and code structure must be maintained. If the structure is maintained by "patching", for example, further modification effort rises exponentially to nightmare proportions.

Therefore, the selection of a design language should permit the introduction of changes without increasing the complexity of the original structure.

Several principles directly support the attainment of this quality. The first is abstraction. By reducing the number of details a designer is required to know and understand, at any particular point in the design, the system becomes more structured and understandable and thus more maintainable. When a structured system has been developed utilizing modules which are independent of each other, modifications should not radically change the overall structure of the system.

Another principle which works in conjunction with modularity is, localization. If modules are created with loose interconnections and cohesive internal elements, then the effects of modification can be limited to a select set of modules.

Two other principles also support modifiability; completeness and confirmability. When both are used together, the system may be easily decomposed, and therefore become easily modifiable.

2. Efficiency

Efficiency evokes the premise that all available resources will be utilized optimally. In the software sense, these resources may be distributed between time and space. Both are obviously somewhat dependent upon the underlying hardware. The final design, in this case, must be amenable to real time events, in order to allow some sense of realism when compared to the actual system. For the actual design of this simulation, space resources should not impose difficulties since implementation is to be made on a large computer system.

The principles which most closely support this quality include completeness and confirmability. Completeness ensures that all important elements are present, permitting fine-tuning of a low-level implementation without affecting higher level modules. Confirmability then allows for testing the "improvements" of this find-tuning, to ensure that the system actually achieves the stated goals of the specifications.

3. <u>Reliability</u>

Reliability is a crucial quality for the HWS system. It is not a quality which can be built in after design is accomplished; it must be instituted from the beginning. In the case of the simulation model, if a failure occurs, degradation should occur gracefully, and not allow fatal side effects. If monitoring of simulated ship sensors is taking place automatically, the loss of one sensor should not necessarily cause the simulation to fail. In this case, manual inputs could be encouraged through exception handling.

All of the principles already discussed which support modifiability and efficiancy also apply to reliability, but for somewhat different reasons. By applying abstraction, and the additional principle of information hiding, only logical operations may be accomplished at any particular level. Similarly, if modularity and localization have been applied in some purposeful manner, then there will be limited interface between the systems modules, further enhancing reliability through reduced complexity.

4. <u>Understandability</u>

In producing the model translation, understandability is probably one of the most desirable qualities. It serves in managing the intricacies of software systems, by

acting as a interlink between a problem definition and an appropriate solution. Understandability is, therefore, fundamental to the other qualities of modifiability, efficiency and reliability.

Every principle discussed thus far also supports understandability. Abstraction extracts essential details, while information hiding suppresses how they are implemented. Modularity and localization then places these details into some well structured design. Then with another principle of uniformity, all similar structures will be of the same logical design. Then with completeness and confirmability, all the necessary information will be present to understand the system.

B. ADA OVERVIEW

Before specifying Ada as the model design language, an overview of the language is needed to see if it will meet the selection criteria. Part of the criteria set forth by the Ada designers was, program maintainability, program reliability, efficiency, and consideration of programming as a human activity. If, in fact, these criteria have been achieved, then at least on the surface, the design criteria has already been met. However, a quick look at the structure of an Ada program might prove to be a better indicator.

Ada is an extremely large language. It's purpose is to be able to respond to important programming issues in practical, real world systems. To facilitate ease of use, an Ada program consists of one or more units, where each unit can be compiled separately. To abstract further, each unit may be composed of any combination of subprograms, tasks and packages.

1. Subprograms

Subprograms in Ada, like other languages, are parameterized units of programming. It may be a procedure or function, and it defines a single action. Functions are utilized as components of an expression, and therefore return some result. Procedures, on the other hand, are called by statements, and return no results. These components seem readily relegated to performing fairly specific duties, such as computing the Uncertainity Ellipse at the lowest level of the simulation model. The subprogram unit readily supports the principles of modularity, abstraction, localization and information hiding.

2. Tasks

The task defines some action that is logically executed in parallel with other tasks. It can be visualized as independent but concurrent operations of program units. This would therefore allow implementation on a multiprocessor, a single processor, or a network of processors. This gives Ada the added attribute of flexibility. With the ability to conduct parallel action, the task structure would be quite amenable to accessing and updating data bases in the simulation model, and for obtaining updates from simulated ship sensors.

3. Packages

Packages are components of a program unit which allow for encapsulation of other components which are logically related. This includes data types, data objects, subprograms, tasks, and even other packages. This allows for the expression and enforcement of a logical abstraction. Packages further support the principle of information hiding. Only the information necessary for the user to

utilize the package is presented. The body or implementation is hidden from the user since he has no need to know how the package is constructed.

4. Other Ada Features

There are several other items in Ada that tend to encourage selection of Ada as the simulation model design language. The first is the exception handling capability. If an error (e.g., divide by zero, buffer overflow) or failure (e.g., peripheral fails), processing should still continue even at reduced capability. Ada permits user defined exceptions, as well as some predefined exception conditions. The exception handler would be ideal for describing situations where simulated ships sensors simulate failure conditions.

The generic program unit is another positive feature for selecting Ada. With this tool, an algorithm can be written to perform the same operation, but on different data types. This feature would be useful for describing situations where single, or even multiple, fields of different data bases are updated.

Globally assigned variables are also provided by Ada. This would be useful in identifying the current state, for menu display purposes, for all modules in the program. However, if needed, packages could localize the effects of these variables. It is doubtful that many global variables would be needed for this simulation, but the option is available. However, extensive use of globally assigned variables is considered poor programming practice.

A final feature is the CLOCK facility. Most programming languages require access to the operating system to obtain time services, but Ada has provided an avenue to obtain time information without defaulting to the operating system. Since the simulation will require the services of a

real time clock, this facility would lead to a more understandable simulation design.

C. ADA AS THE SYSTEM DESIGN LANGUAGE

It is apparent that Ada holds all those qualities needed in designing a system. Therefore, Ada is recommended as the system design language.

Reference 3 established a structure for the program design as depicted in Figure 4.1. In presenting this structure, two modifications were made. First, the Launcher Missile Status was moved under the Update Module, to accommodate grouping of similar tasks. Secondly, the Threat Display was moved up one level for the same reason. The overall structure continues to remain the same as that shown in Figures 2.1 through 2.4.

Appendix F presents a generalized, top-down type approach in converting to a system design language. It's purpose is to give a rough presentation of several of the structure modules of figure 4.1. Although the semantics have not been checked, the syntax has been verified. This layout should assist in the actual conversion from the specifications to the system design language.

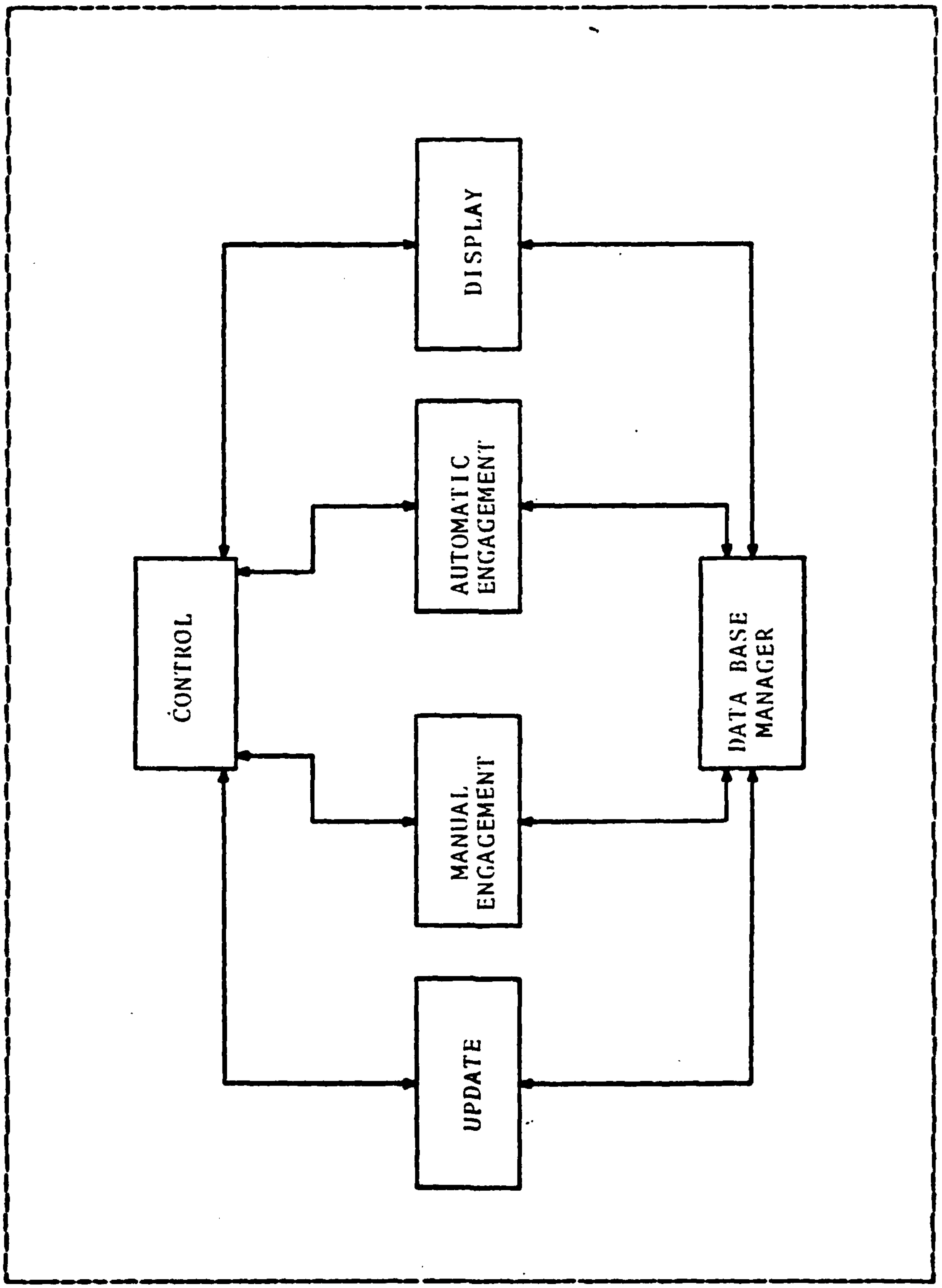


Figure 4.1 Program Design Structure.

V. CONCLUSIONS AND RECOMMENDATIONS

In developing a design for a simulation model, it is often easy to loose track of the original problem. This thesis has attempted to establish a design to validate the specifications for the HSCLCS update, identify problem areas and necessary changes to those specifications. This design should ultimately lead to a verification of the system requirements.

Although the intended purpose of the simulation is not to create a prototype of the HSCLCS, it can be used to develop such a prototype and, in addition, if properly designed, be used as a valuable training device. This training can be applied as a method of providing access to the type of information an operator can expect to encounter, or provide a reasonable facsimile of a real time HARPCON engagement.

In developing this design, several items appeared to be the most significant. First was the decision as to whether the problem could best be solved by a simulation. Reference 3 provided an analysis to help make this decision in the affirmative. Next, was the issue of narrowing down the boundaries of the real system which should be simulated. As the model design progresses, the need to modify these decisions will undoubtedly be required. This will entail removing some items, since normally most designers tend to incorporate too much vice too little. This design will surely support this conclusion.

The model specification was another important issue that this thesis had to resolve. To this end, several concepts proved themselves invaluable. The first was abstraction. This allowed the identification of those features of the

real system which were essential in the model. Then, by utilizing the concept of modularity, each primary feature was specified independent of the specifications of any others.

A final idea that was utilized at the model specification stage, was to avoid restricting the implementor. If the specification contains to much detail, an undue burden may be placed on the actual implementation. This can produce a model that does not accurately reflect the system it is intended to represent.

A. RECOMMENDED FOLLOW-ON WORK

The author recommends exploring the following areas in further support of the HSCLCS improvement and simulation model design methodology:

- Research and develop the algorithm for the uncertainity ellipse to support the HSCLCS simulation model.
- Research and develop the algorithm for the probability of acquisition to support the HSCLCS simulation model.
- Develop a graphics package in support of the HSCLCS simulation model.
- Investigate different computer systems for implementation of a final design.
- Convert the simulation model design into the System Design Language using Appendix F as a guide.
- Research whether of aspects in this simulation which might be transferable to simulation model development of other cruise missiles and their follow-ons.
- Study the feasibility of utilizing Ada as the program design language, in addition to the system design language, for the HSCLCS simulation model.

APPENDIX A

GLOSSARY

Data Base - A file of interrelated data stored together to serve one or more applications and that remains independent of programs using the data.

Data Structure - Dictates the organization, methods of access, degree of associativity and processing alternatives for information.

Embedded System Program - A computer program that is part of some larger entity and essential to the operations of that system. For example, the timer on a washing machine or the guidance system in a missile may have computer programs which are considered to be embedded.

Function - Name given to one or more statements that perform a specific task. Results in a value being assigned to its name upon execution of that specific function.

Information Hiding - Specification and design of modules so that information (procedure and data) contained within a module are inaccessible to other modules that have no need to know the information.

Interface - Communications between modules governed by a set of assumptions one module makes about another.

Module - A seperately addressable element with a single coherent purpose.

Modular Design - A logical partitioning of software into elements that perform specific functions or subfunctions.

Navy Tactical Data System - A communications link between different platforms (e.g., surface ships and aircraft, submarines and surface ships). Real time information is passed via this communication link.

Package - A program unit specifying a collection of related entities such as constants, variables, types, and subprograms. The visible part of the package contains the entities that may be used from outside the package. The private part of the package contains structural details that are irrelevant to the user of the package but that complete the specification of the visible entities. The package body contains the implementation of the subprograms or task (possibly other packages) specified in the visible part.

Packaging - Alludes to the techniques used to assemble software for specific processing environment or to ship software to a remote location.

Probability of Acquisition - Calculated probability of seek-head acquisition of intended target based upon information available.

Subordinate Module - A module controlled by another module.

Subprogram - An executable program unit, possibly with parameters for communication with its point of call. A subprogram declaration specifies the name of the subprogram and its parameters; a subprogram body specifies its execution. A subprogram may be a procedure, which performs an action, or a function, which returns a result.

System - A collection of elements related in a way that allows accomplishment of some tangible objective.

Task - A program unit that may operate in parallel with other program units. A task specification establishes the name of the task and the names and parameters of its entries; a task body defines its execution. A task type is a specification that permits the subsequent declaration of any number of similar tasks. A task is said to depend upon the unit in which it is declared (subprogram body, task body, or a library package body). A unit is not left until all dependent tasks are terminated. A task is completed if

it is waiting at the end of its body for any dependent tasks or is aborted but not yet terminated. A completed task cannot be called. A terminated task is, in a sense, the same as a dead task in that it is no longer active.

Uncertainty Ellipse - A probability associated with a track (or target) that its position is within a given geographical location.

APPENDIX B

ACRONYMS AND ABBREVIATIONS

BIT	-	Built In Test
BOL	-	Bearing Only Launch
BRG	-	Bearing
CML	-	Cannister Missile Launcher
CPU	-	Central Processing Unit
DCU	-	Data Conversion Unit
DPC	-	Data Processor Computer
GDS	-	Graphics Display System
HCC	-	HARPOON Control Console
HSCLCS	-	HARPOON Shipboard Command-Launch Control Set
HWS	-	HARPOON Weapons System
NTDS	-	Naval Tactical Data System
OLS	-	On-Line-Sizing
RAM	-	Random Access Memory
RBL	-	Range and Bearing Launch
RNSH	-	Royal Navy Sublaunched HARPOON
STOT	-	Simultaneous Time-on-Target
UV-EPROM	-	Ultra-Violet Eraseable Programmable Read-Only Memory
WCIP	-	Weapon Control-Indicator Panel

WCS - Weapons Control System

ZULU - Greenwich Mean Time

APPENDIX C

HSCLCS ENGAGEMENT PLANNING/OPERATIONAL EMPLOYMENT FUNCTIONS

Operational Data/Information	Requirement Baseline	Requirement Design Goal
1a. Surface Contact Position (range/bearing). The use of bearing line in addition to the 1b requirement reduces the number of displayed surface contacts by two per bearing line.	10	20 (min)
-Designated Target Target Category and Classification Displayed.	X	
-Unintended Target(s) Target Category and Classification Displayed.	X	
1b. Surface Contact/Bearing Line	1	3 (min)
2. Own Ship Position	X	
3. Air Contact Position	1	3 (min)
4. 3rd Party Targeting Data Source Designation WCIP shall resolve target position based on range and bearing input from 3rd party or bearing lines from 3rd parties or own ship.	2	3 (min)
-Manual Entry of Bearing Lines	X	
-Manual Entry of Range and Bearing	X	
5. Target Classification		

-Large (default)	X
Larger than a patrol boat.	
-Small	X
Patrol boat or smaller.	
6. Contact/Track Course Direction Indicator	
Program automatically compensates for own ship's motion.	
-Direction Indicator	X
-Dead Reckoning (Own Ship Only)	X
7. Contact/Track Targeting Data Source	
-Manual Input	X
With appropriate data source error; includes 3rd party.	
-Automatic Input	
-NTDS	X
-RADAR	X
-SONAR	X
-EW/ESM	X
-Target Designation System	X
8. Wind Parameters (relative)	
-Speed	
-Actual	X
Manual input.	
-Default value	X
-Direction	
-Actual	X
Manual input.	
-Default value	X
9. Temperature	
-Actual	X
Manual input.	
-Default value	X

10. Precipitation	
-Yes	X
Manual input.	
-No (default value)	X
11. Operator Cues/Lockouts	
-Launch Inhibited (lockouts/cue)	X
All launch inhibits except roll/ pitch cutout.	
-Missile Ready (cue)	X
-Data Age (cue)	X
Target and environmental data.	
-Missile Launch Status (cue)	
-Cell/Rail Empty (missile away)	X
-Missile Dud Declaration	X
-New Contact/Track to be Input (cue)	X
-Illegal Action (lockout/cue)	X
12. Time/Clock	
-ZULU Time	X
Start clock: Automatic entry via NTDS Interface and/or manual entry.	
-Time on Target	X
Manual entry.	
-Time of Launch	X
Computation.	
-Countdown	X
Includes Time-to-Fire and Time-to-Impact.	
13. Loadout Status/Missile Variant Identification	
-Baseline/Block I Tactical Missile (RGM-84A)	X
-Royal Navy Submarine HARPOON (RGM-84B)	X

When reconfigured for surface launch.		
-Block IB Tactical Missile (RGM-84C)	X	
-Block IC Tactical Missile (RGM-84D)	X	
-Supplemental Identification (manual entry: info from loadout logbooks of hybrid/nonstandard seeker-MGU combinations).	X	
-Training All-Up Round (RTM-84A/C/D and RNSH)	X	
14. Missile In-Flight Tracks	X	
15. Up to 180 degree Off-Axis Launch	X	

Operational Selections

1. Reference System		
-True Target Bearing/Relative Target Range Top of display is north.	X	
-NTDS Grid		X
-Geographic (latitude & longitude)		X
2. Planned Missile Flight Path Software to ensure that no flight path may be selected which could result in the acquisition of own ship.	3 WPS	
3. Search Mode Selection		
-On Line Sizing (default) w/Manual Override On Line Sizing shall be automatically selected if RBL or BOL are	X	

not selected.

-Range and Bearing Launch (RBL) X

RBL pattern size shall be a function of total flight path (range traveled to target).

-Bearing Only Launch (BOL) X

4. Selectable Search Pattern Expansion (0 - 360 degrees) X

For RGM-84D missile only, applies to RBL mode and On-Line-Sizing (OLS) which results in an RBL search pattern.

-Normal Center Expansion X

For RGM-84A/RGM-84B/RGM-84C missiles; default for RGM-84D missile.

5. Enable and Destruct Ranges BOL X

Default values or manual entry (ranges not supplied over NTDS interface).

6. High Altitude Hold

RGM-84D only.

-No Entry; Default X

The High Altitude Hold default range not to interfere with search initiation and not to exceed 10nm; i.e., High Altitude Hold range is set to the minimum of 10nm or range to search initiation.

-Manual Entry X

The selected High Altitude Hold range must be less than the range to search initiation.

7.	Presearch Fly-Out		
	-Sea Skim (RGM-84D only)	X	
	Default mode - Presearch Fly-Out is set to sea skim altitude following the High Altitude Hold.		
	-Manual Entry	X	
	Presearch Fly-Out at normal HARPOON run-in altitude as used in current HSCLCS.		
8.	Terminal Attack Mode (RGM-84D only)		
	-Sea Skim (default)	X	
	-Pop-up	X	
	Default override by manual selection of pop-up, "SMALL TARGET" designation by NTDS, or when "SMALL TARGET" is entered manually.		
9.	Missile Assignment for Engagement Planning	X	
	Manual entry.		
10.	Multi-Missile Engagement of Designated Target.	4	8
	Baseline: Up to 4 missiles from a single launcher. (Note: Single launcher includes TARTAR and ASROC). Design Goal: Up to 8 missiles from 2 CML's.		
	-Salvo Missiles Against One Target For Simultaneous Arrival (STOT Salvo).	X	
	Operator-planned engagement.		
	-Salvo Against Up to Four Targets (single airpoint) From One Launcher For Simultaneous Arrival (STOT	X	

Salvo).
Same aimpoint and a different RBL search expansion for each RGM-84D missile in order to distribute salvoed missiles among the targets in a formation.

-Ripple Salvo as per current HSCLCS CML Configuration.	X
-Quick Reaction/Preprogrammed STOT Salvo. Modified HSCLCS automatically will calculate and enter a different waypoint for each RGM-84D missile in a quick reaction salvo for simultaneous time-on-target (STOT).	X
11. Background data and sector data request. Usable with NTDS interface only.	X

ENGAGEMENT DISPLAYS

1. Contact/Track Uncertainity Ellipse	
-Designated Surface Target	X
-Unintended Targets If selected by operator.	X
2. Predicted Time-on-Target	X
3. Probability of Acquisition Numerical value.	
-Designated Targets	X
-Unintended Targets If selected by operator.	X
4. Seeker Search Pattern Outline For selected search mode.	X

5.	Missile Flight Path For all selected missiles.	X
6.	Booster Drop Zone	X
7.	Missile Power Application Warning	X

OTHER

1.	Test/Maintenance	
	-Missile BIT Results	
	-Go/No-Go Indication	X
	-Failure Status Code	X
	-HSCLCS BIT Results	
	-Go/No-Go Indication	X
	-Failure Status Code	X
2.	Training Mode Inherent capability provided by system design. Design to utilize data from NTDS and/or external training support devices via an RS 232 serial interface.	
	-Contact/Track Location (actual or simulated).	
	-Off Board Source/NTDS	X
	-Own Ship Sensors/NTDS	X
	-Manual Input	X
	-Own Ship Position (actual or simulated).	X
	-Training Scenario Parameters	
	-Environmental Conditions	X
	-Operational Planning Selections	X
3.	Data Extract Design to be compatible with an RS 232 serial interface to provide for	

data storage/display in off-line devices (e.g., tape cassette recorder).

-Target/Targeting Data	X
-Missile Initialization Data	X
-BIT Results	X

4. Major Display Features

-Variable Range Scale	X
16K-, 32K-, 64K-, 128K-, 192K-, or 256K-yard radius. The 256K-yard is the default scale.	
-Offset	X
-Zoom	X
8K-, 16K- or 32K-yard radius.	
-Special Symbols	X
-Cursor, with Bearing/Range readout	X
Manually controlled.	

APPENDIX D

DATA BASE DESCRIPTIONS

This appendix contains the data base descriptions to be used in the simulation model. These data bases include:

1. Environmental Data Base
2. Launcher and Missile Status Data Base
3. Menu/State Data Base
4. Ship Parameter Data Base
5. Threat Data Base
6. Engagement Data Base
7. Track Data Base

1. Data Base Name: ENVIRONMENTAL DATA BASE

2. Purpose: This data base contains the current state of weather; visibility, sea state, winds, rain, etc.

3. Data Base Users:

   a. Write Access: Environment Data Base Manager

   b. Read Access: Environment Display
      Plan Engagement

4. Data Base Elements: Visibility
   Sea state
   Wind-direction and speed
   Relative Humitity
   Temperature
   Barometric Pressure
   Precipitation

5. Operation on Data Base: Update (manual or automatic)
   - Add
   - Delete
   - Change all elements
   Display

1. Data Base Name: LAUNCHER AND MISSILE STATUS DATA BASE

2. Purpose: This data base will keep track of the number and type of missiles available for launch. The data base will be updated by feedback from the launcher.

3. Data Base Users:

   a. Write Access: Launcher Missile Status

   b. Read Access: Launcher Missile Assignment
   Plan Engagement
   Launcher Missile Status Display

4. Data Base Elements: Launcher Number
   Cannister Number
   Missile type
   Launch Inhibit

5. Operations on Data Base: Update - Add
   - Delete
   - Change all elements
   Display

1. Data Base Name: MENU/STATE DATA BASE

2. Purpose: This data base will contain the menus for program operation and also provide the states allowable from any operation. That is it will provide the menus necessary to access all aspects of the program.

3. Data Base Users:

   a. Write Access : None

   b. Read Access: Display Module

4. Data Base Elements: Undetermined at this time

5. Operations on data Base: Operator can select desired menu item

1. Data Base Name : SHIP PARAMETER DATA BASE

2. Purpose: This data base will maintain all pertinent information pertaining to one's own ship. The design allows for this information to be input manually or automatically.

3. Data Base Users:

   a. Write Access: Ship Parameter Data Base Manager

   b. Read Access: Update Track
      Plan Engagement
      Ship Parameter Display

4. Data Base Elements: Course
   Speed
   Latitude
   Longitude

5. Operations on Data Base: Update - Change all elements
   Display

1. Data Base Name: THREAT DATA BASE

2. Purpose: This data base is to contain a list of hostile surface vessels by name and class. Associated with each class of vessel will be the weapons platform, ECM capabilities, and optimum engagement plan for attacking that vessel. (The security of this information must be considered when designing the Threat Display and Threat Data Base Manager modules).

3. Data Base Users:
   a. Write Access: Threat Data Base Manager
   b. Read Access: Threat Display
      Analyze Threat Data

4. Data Base Elements: Ship Name
   Nationality
   Ship Class
   Weapons
   ECM Equipment
   Engagement Plan Recommended

5. Operations on Data Base: Update - Add
   - Delete (Permissions)
   - Change all Elements (Permissions)

   Display

1. Data Base Name: ENGAGEMENT DATA BASE

2. Purpose : This data base will contain a track name and associated engagement plans for that track. The engagement plan may be generated automatically by the computer or manually.

3. Data Base Users:

   a. Write Access: Calculate Probability of Acquisition

   b. Read Access : Engagement Plan Display

4. Data Base Elements: Track name
   Type of engagement plan (manual or automatic)
   Number of missiles to fire
   Sequence of firing missile(s)
   Type of missile to use
   Flight path
   Waypoints

5. Operations on data base: Update
   Display

1. Data Base Name: TRACK DATA BASE

2. Purpose: This data base will contain the position of all tracks and pertinent information pertaining to the track.

3. Data Base Users:

   a. Write Access: Delete Track
      Add Track
      Course and Speed Update
      Bearing, Range and Position Update

   b. Read Access: Display Track Data
      Plan Engagement

4. Data Base Elements: Type track (friend or foe)
   Class of vessel
   Bearing
   Range
   Position (lattitude and longitude)
   Course
   Speed

5. Operations on Data Base: Update - add
   - delete
   - change bearing range or position
   Display

APPENDIX E

MODULE DESCRIPTIONS

This appendix contains the module descriptions of the modules shown in Figure 2.1 through Figure 2.4. These modules include:

1 -	0	Control
2 -	1	Process Input
3 -	1.1	Ship Parameter Data Base Manager
4 -	1.2	Environmental Data Base Manager
5 -	1.3	Threat Data Base Manager
6 -	2	Convert Coordinates
7 -	2.1	Type Track
8 -	2.1.1	Delete Track
9 -	2.1.2	Update Track
10 -	2.1.2.1	Course and Speed Update
11 -	2.1.2.2	Bearing, Range, and Position Update
12 -	2.1.3	Add Track
13 -	3.1	Launcher and Missile Assignment
14 -	3.1.2	Launcher and Missile Status
15 -	3.2	Plan Engagement
16 -	3.2.1	Plan Engagement Data Base Manager
17 -	3.2.2	Engagement Data
18 -	3.2.2.1	Threat Data
19 -	3.2.3	Probability of Acquisition
20 -	3.2.3.1	Uncertainty Ellipse
21 -	4	Display
22 -	4.1	Menu Display
23 -	4.2	Launcher and Missile Status Display
24 -	4.3	Environmental Display
25 -	4.4	Engagement Display
26 -	4.4.1	Threat Display
27 -	4.4.2	Automatic Engagement Display
28 -	4.4.2.1	Graphics Display
29 -	4.4.3	Manual Engagement Display
30 -	4.5	Ship Parameter Display
31 -	4.6	Track Display

1. Module Name: CONTROL, NUMBER 0

2. Module Purpose: The Control module calls all other modules and determines the program flow.

3. Subordinate Modules: Process Input (1)
Launcher Missile Assignment (3.1)
Plan Engagement (3.2)
Display (4)

4. Objects Used by the Module: Manual inputs

5. Operations Module Performs: Selection of subordinate modules to perform program operation.

1. Module Name: PROCESS INPUT, NUMBER 1

2. Module Purpose: Selects subordinate module to update corresponding data bases.

3. Subordinate Modules: Ship Parameter Data Base Manager (1.1)
Environmental Data Base Manager (1.2)
Convert Coordinates (2)
Threat Data Base Manager (1.3)

4. Objects Used by the Module: Manual Inputs to update modules

5. Operations Module Performs: Selects appropriate subordinate module to update corresponding data base.

1. Module Name: SHIP PARAMETER DATA BASE MANAGER, NUMBER 1.1

2. Module purpose: Update the Ship Parameter Data Base by either manual or automatic means.

3. Subordinate Modules: None

4. Objects Used by the Module: Ship parameter input data

5. Operations Module Performs: Update of the Ship Parameter Data Base.

1. Module Name: ENVIRONMENTAL DATA BASE MANAGER, NUMBER 1.2

2. Module Purpose: Update the Environmental Data Base by either manual or automated means.

3. Subordinate Modules: None

4. Objects Used by the Module: Environmental input or update data.

5. Operations Module Performs: Update of the Environmental Data Base.

1. Module Name: THREAT DATA BASE MANAGER, NUMBER 1.3

2. Module Purpose: Update the Threat Data Base by either manual means or through the use of a standard chip that can be periodically updated and sent to all ships with HARPOON capability.

3. Subordinate Modules: None
4. Objects Used by the Module: Data used to update the Threat Data Base.

5. Operations Module Performs: Update of the Threat Data Base.

1. Module Name: CONVERT COORDINATES, NUMBER 2

2. Module Purpose: To convert all inputs to update track data to common coordinates. The inputs can be manual, from own ship's tracking equipment, or from a NTDS link from other platforms.

3. Subordinate Modules: Type Track (2.1)

4. Objects Used by the Module: Information used to update the Track Data Base, bearing, range and position.

5. Operations Module Performs: All sources of input for the Track Data Base are converted to common coordinates whether they are manual, NTDS or from another source.

1. Module Name: TYPE TRACK, NUMBER 2.1

2. Module Purpose: Type Track determines if the track is to be deleted from the database, added to the database, or some parameters of a present track are to be altered. These actions are performed by selecting the appropriate subordinate module.

3. Subordinate Modules: Delete Track (2.1.1)
   Update Track (2.1.2)
   Add Track (2.1.3)

4. Objects Used by the Module: Classification of type update to be performed; addition, deletion or alteration to the Track Data Base.

5. Operations Module Performs: Selection of delete, update, or add track subordinate modules based on track type.

1. Module Name: DELETE TRACK, NUMBER 2.1.1

2. Module Purpose: To eliminate tracks from the data base that the operator determines are no longer useful.

3. Subordinate Modules: None

4. Objects Used by the Module: None

5. Operations Module Performs: Track identified to the Delete Track module is removed from the Track Data Base.

1. Module Name: UPDATE TRACK, NUMBER 2.1.2

2. Module Purpose: To update the information contained on tracks in the Track Data Base.

3. Subordinate Modules: Course and Speed (2.1.2.1)
Bearing and Range (2.1.2.2)

4. Objects Used by the Module: Bearing and range from a fixed point.
Elapsed time.
Own ship course and speed.
Target course and speed.

5. Operations Module Performs: Determines which subordinate module is to be called to perform the desired update.

1. Module Name: COURSE AND SPEED UPDATE, NUMBER 2.1.2.1

2. Module Purpose: To update the course and speed information on each track contained in the Track Data Base.

3. Subordinate Modules: None

4. Objects Used by the Module: Own ship course and speed.
Ellapsed time.
Bearing and range from a fixed point.
NTDS link information.

5. Operations Module Performs: Determines course and speed of tracks based on bearing/range and ellapsed time when entered manually and updates Track Data Base. With automated track information available (e.g., NTDS) module updates Track Data Base with the given course and speed.

1. Module Name: BEARING, RANGE, AND POSITION UPDATE, NUMBER 2.1.2.2

2. Module Purpose: To update the bearing/range and position (Latitude and Longitude) information on each track contained in the Track Data Base.

3. Subordinate Modules: None

4. Objects Used by the Module: Own ship sensor information.
NTDS link information

5. Operations Module Performs: Determines bearing/range and position of tracks based on own ship sensor information and own ship position, when entered manually and updates Track Data Base. With automated track information available (e.g., NTDS) module updates Track Data Base with the given bearing/range and position.

1. Module Name: ADD TRACK, NUMBER 2.1.3

2. Module Purpose: To allow new tracks to be input into the Track Data Base.

3. Subordinate Modules: None

4. Objects Used by the Module: Course, speed, bearing, range, position, identification of track (friend or foe and ship class).

5. Operations Module Performs: Permits the addition of new tracks into the Track Data Base.

1. Module Name: LAUNCHER MISSILE ASSIGNMENT, NUMBER 3.1

2. Module Purpose: Allow the operator to bypass the engagement planning capabilities of the computer system and simply select and launch the desired missiles.

3. Subordinate Modules: Launcher and Missile Status

4. Objects Used by the Module: Inputs from operator identifying which launcher and missiles to be fired in which bearing, range and waypoints if desired.

5. Operations Module Performs: The Launcher Missile Assignment module allows the operator to manually select a launcher and missile to be fired in a given direction similar to the present capabilities of the HARPOON Weapons System. The automated engagement planning functions of this program are bypassed.

1. Module Name: LAUNCHER AND MISSILE STATUS, NUMBER 3.1.2

2. Module Purpose: To provide current information on what launchers (port and starboard) are ready to fire and which and what type missiles are ready for firing.

3. Subordinate Modules: None

4. Objects Used by the Module: Which launchers (port and starboard) are ready to fire. Which and what type missiles are ready for firing.

5. Operations Module Performs: The Launcher and Missile Status module receives automated inputs from each launcher on the status and type of all missiles. This information is used to update the Launcher and Missile Status Data Base. When queried by either the Launcher and Missile Assignment module or the Engagement Data module, the Launcher and Missile Status module can return the status of launchers and missiles.

1. Module Name: PLAN ENGAGEMENT, NUMBER 3.2

2. Module Purpose: To determine the optimum engagement plan for a given target.

3. Subordinate Modules: Plan Engagement Data Base Manager (3.2.1)
   Engagement Data (3.2.2)
   Probability of Acquisition (3.2.3)

4. Objects Used by the Module: Which track to plan the engagement for.

5. Operations Module Performs: The Plan Engagement module is the heart of this software program. This module determines an optimum engagement plan for desired targets. The targets that have an engagement plan computed can be identified either by a selective manual input or can be automatically generated for all contacts that are classified hostile (This latter process would require a slight modification to the design; an input to the Plan Engagement must be received from the type track module). Through access to the Threat Data Base, Launcher Missile Status Data Base the optimum plan for engaging a selected target can be computed. The functions performed by this module can greatly assist the Tactical Action Officer in the performance of his duties.

1. Module Name: PLAN ENGAGEMENT DATA BASE MANAGER, NUMBER 3.2.1

2. Module Purpose: To update the Engagement Plan Data Base.

3. Subordinate Modules: None

4. Objects Used by the Module: Engagement plan as generated by the Engagement Plan Module.

5. Operations Module Performs: The Plan Engagement Data Base Manager enters all engagement plans that the Plan Engagement module generates into a Plan Engagement Data Base. This way a list of the engagement plans for all applicable targets is kept current in a data base.

1. Module Name: ENGAGEMENT DATA, NUMBER 3.2.2

2. Module Purpose: The Engagement Data module supplies the data needed by the Plan Engagement module to generate the Engagement Plan.

3. Subordinate Modules: Launcher and Missile Status (3.1.2)
Threat Data (3.2.2.1)

4. Objects Used by the Module: Which launcher and missiles are ready to fire.
All pertinent information on hostile ship class, weapons, ECM equipment and best strategy for attack contained in the Threat Data Base.

5. Operations Module Performs: The Engagement Data module coordinates the passing of all data base information needed to generate an engagement plan to the Plan Engagement module. In this design, that information is contained in the Launcher and Missile Status module and the Threat Data module.

1. Module Name: THREAT DATA, NUMBER 3.2.2.1

2. Module Purpose: To provide the information contained in the Threat Data module to the Engagement Data module when requested.

3. Subordinate Modules: None

4. Objects Used by the Module: All elements contained in the Threat Data Base, ship class, weapons, platform, ECM capability and best plan for attack.

5. Operations Module Performs: The Threat Data module provides the Engagement Plan module with all of the information that is contained in the Threat Data Base. This information is then used to determine the optimum engagement plan.

1. Module Name: PROBABILITY OF ACQUISITION, NUMBER 3.2.3

2. Module Purpose: To determine what the probability is that if a missile is launched at a given target that the missile can acquire and hit that target.

3. Subordinate Modules: Uncertainty Ellipse (3.2.3.1)

4. Objects Used by the Module: The figure generated by the Uncertainty Ellipse module. Type missile to be launched and search pattern. Type target to be attacked and its physical characteristics and ECM capabilities.

5. Operations Module Performs: The Probability of Acquisition module uses the type missile fired, range to target, and target characteristics to generate the probability that the missile can acquire and hit the given target. This information along with the value obtained from the Uncertainty Ellipse module is then passed to the Plan Engagement module where they become elements of the engagement plan maintained in the Engagement Plan Data Base.

1. Module Name: UNCERTAINTY ELLIPSE, NUMBER 3.2.3.1

2. Module Purpose: To determine the probability that a given missile, or set of missiles, fired at a specific target will sink that target.

3. Subordinate Modules: None

4. Objects Used by the Module: Number of missiles to be fired.
Probability of acquisition of the target.
Lethal capability of missiles fired.

5. Operations Module Performs: The Uncertainty Ellipse module takes the number and capabilities of missiles fired and combines these values with the probability of acquisition to generate the probability of a target kill. The Uncertainty Ellipse module can generate ellipses with assigned probabilities stating that total destruction of the target will occur if it is within one ellipse, 50% disability of the target will occur if it is within a second ellipse, etc. These ellipses will account for the fact that hostile target positions may not be completely accurate.

1. Module Name: DISPLAY, NUMBER 4

2. Module Purpose: To call subordinate modules as necessary to generate required displays.

3. Subordinate Modules: Menu Display (4.1)
Launcher and Missile Status Display (4.2)
Environmental Display (4.3)
Engagement Display (4.4)
Ship Parameter Display (4.5)
Track Display (4.6)

4. Objects Used by the Module: Display requests.

5. Operations Module Performs: The Display module calls the required modules to generate display as necessary.

1. Module Name: MENU DISPLAY, NUMBER 4.1

2. Module Purpose: To access the Menu/State Data Base and display the required menu when called and keep track of the state of the program.

3. Subordinate Modules: None

4. Objects Used by the Module: Information contained in the Menu/State Data Base.

5. Operations Module Performs: The Menu Display module will access the Menu/State Data Base and provide the necessary menu for display when prompted by the Display module. The Menu Display module will also keep track of the state of the program, that is what menus can be displayed given that the state of the program exists now with a current menu.

1. Module Name: LAUNCHER AND MISSILE STATUS DISPLAY, NUMBER 4.2

2. Module Purpose: To access the Launcher and Missile Status Data Base and provide a display of the information contained in that data base.

3. Subordinate Modules: None

4. Objects Used by the Module: Information contained in the Launcher and Missile Status Data Base.

5. Operations Module Performs: The Launcher and Missile Status Display module will display the information contained in the Launcher and Missile Status Data Base when prompted by the Display module.

1. Module Name: ENVIRONMENTAL DISPLAY, NUMBER 4.3

2. Module Purpose: To access the Environmental Data Base and provide a display of the information contained in that data base.

3. Subordinate Modules: None

4. Objects Used by the Module: Information contained in the Environmental Data Base.

5. Operations Module Performs: The Environments Display module will display the information contained in the Environmental Data Base when prompted by the Display module.

1. Module Name: ENGAGEMENT DISPLAY, NUMBER 4.4

2. Module Purpose: To graphically display the flight path of missiles that are to be flown against a set target. Threat data on the target will also be displayed. The engagement play will have the capability to be superimposed over the general track display.

3. Subordinate Modules: Threat Display (4.4.1)
   Automatic Engagement (4.4.2)
   Manual Engagement (4.4.3)

4. Objects Used by the Module: Threat Data Base information.
   Engagement Plan Data Base information.
   Manual inputs for an engagement plan.

5. Operations Module Performs: The Engagement Display module calls upon subordinate modules to provide the operator a display of the computer generated engagement plan constructed by the operator. In both cases, the threat data pertinent to the displayed target can also be shown.

1. Module Name: THREAT DISPLAY, NUMBER 4.4.1

2. Module Purpose: To access the Threat Data Base and provide a display of the information contained in that data base.

3. Subordinate Modules: None

4. Objects Used by the Module: The information contained in the Threat Data Base.

5. Operations Module Performs: The Threat Display module will display the information contained in the Threat Data Base when prompted by the Engagement Display module.

1. Module Name: AUTOMATIC ENGAGEMENT DISPLAY, NUMBER 4.4.2

2. Module Purpose: To graphically display the engagement plan that was generated by the Plan Engagement module and stored in the Engagement Plan Data Base.

3. Subordinate Modules: Graphics Display (4.4.2.1)

4. Objects Used by the Module: Information contained in the Engagement Plan Data Base.

5. Operations Module Performs: The Automatic Engagement Display module provides a graphical representation of the engagement representation of the engagement plan as contained in the Engagement Plan Data Base. All missile trajectories and waypoints will be depicted with associated missile fire times and arrive over the target time. The uncertainity ellipses will also be generated along with the probability of acquisition of the target.

1. Module Name: GRAPHICS DISPLAY, NUMBER 4.4.2.1

2. Module Purpose: To provide graphics capabilities to the Automatic Engagement Display module and the Manual Engagement Display module.

3. Subordinate Modules: None

4. Objects Used by the Module: Engagement Plan Data Base information. Manually input engagement plan.

5. Operations Module Performs: The Graphics Display module provides the graphics capabilities necessary to the Automatic Manual Engagement Display module to accurately portray their given engagement plans.

1. Module Name: MANUAL ENGAGEMENT DISPLAY, NUMBER 4.4.3

2. Module Purpose: To provide the operator the capability to manually input an engagement plan for attacking a given target.

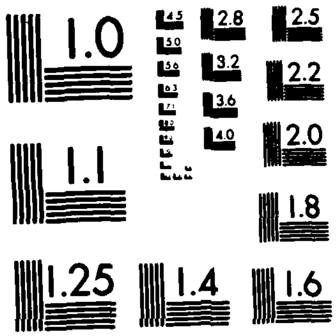
3. Subordinate Modules: Graphics Display (4.4.2.1)

4. Objects Used by the Module: Information contained in the Threat Data Base.

5. Operations Module Performs: The Manual Engagement Display module allows the operator to manually input his own engagement plan for a given target. Once this information is graphically input to the display, it can be transferred to the Engagement Plan Data Base where it can be programmed to the missiles like an automatically generated plan.

AD-A138 654 PRELIMINARY DESIGN FOR HARPOON SHIPBOARD COMMAND-LAUNCH CONTROL SET SIMULATION(U) NAVAL POSTGRADUATE SCHOOL MONTEREY CA J R ESCHLIMAN DEC 83 2/2

UNCLASSIFIED F/G 16/4.2 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

1. Module Name: SHIP PARAMETERS DISPLAY, NUMBER 4.5

2. Module Purpose: To access the Ship Parameter Data Base and provide a display of the information contained in that data base.

3. Subordinate Modules: None

4. Objects Used by the Module: Information contained in the Ship Parameter Data Base.

5. Operations Module Performs: The Ship Parameter Display module will display the information contained in the Ship Parameter Data Base when prompted by the Display module.

1. Module Name: TRACK DISPLAY, NUMBER 4.6

2. Module Purpose: To access the Track Data Base and provide a continuous display of all tracks beings maintained in that data bask.

3. Subordinate Modules: None

4. Objects Used by the Module: Information contained in the Track Data Base.

5. Operations Module Performs: The Track Display module will continuously display all tracks maintained in the Track Data Base. These tracks will be constantly updated as the Track Data Base is updated. The symbology and method presentation of the tracks should closely coincide with NTDS displays.

APPENDIX F

SYSTEM DESIGN USING ADA

```
package UPDATE is
   type NBR_LAUNCHERS;
   type MAXSPEED;
   LAU_NBR : integer range 1..NBR_LAUNCHERS;
   CAN_NBR : integer range 1..7;
   MISS_type : integer range 1..7;
   LAU_INHIBIT : boolean;
   COURSE : integer range 0..359;
   SPEED : integer range 0..MAXSPEED;
   type LATITUDE is record
        DEGREES : integer range -90..90;
        MINUTES : integer range 0..60;
        SECONDS : integer range 0..60;
     end record;
   type LONGITUDE is record
        DEGREES : integer range -180..180;
        MINUTES : integer range 0..60;
        SECONDS : integer range 0..60;
     end record;
   VISIBILITY : integer range 0..30;
   SEA_STATE : integer range 0..10;
   WIND_DIRECTION : integer range 0..359;
   WIND_SPEED : integer range 0..100;
   RELATIVE_HUMIDITY : integer range 0..100;
   TEMPERATURE : integer range -100..130;
   BAROMETRIC_PRESSURE : integer range 900..1100;
   PRECIPITATION : boolean;
   HOSTILE_NAME : string(1..12);
   SHIP_CLASS : string(1..9);
   NATIONALITY : string(1..10);
   WEAPONS : string(1..50);
   ECM_EQUIPMENT : string(1..50);
   ENGAGEMENT_METHOD : string(1..50);
   task LAUNCHER_MISSILE_STATUS is
     entry UPDATE_LAU(LAUNCHER_NUMBER : in LAU_NBR;
        CANNISTER_NUMBER : in CAN_NBR;
        MISSILE_type : in MISS_type;
        LAUNCHER_INHIBIT : in LAU_INHIBIT);
   end LAUNCHER_MISSILE_STATUS;
   task SHIP_PARAMETER is
     entry UPDATE_COURSE(SHIP_COURSE : in COURSE);
     entry UPDATE_SPEED(SHIP_SPEED : in SPEED);
     entry UPDATE_LATITUDE(SHIP_LATITUDE : in LATITUDE);
     entry UPDATE_LONGITUDE(SHIP_LONGITUDE : in LONGITUDE);
   end SHIP_PARAMETER;
   task ENVIRONMENT is
     entry UPDATE_VISIBILITY(VIS : in VISIBILITY);
     entry UPDATE_SEA_STATE(SEA : in SEA_STATE);
     entry UPDATE_WIND_DIR(WINDDIR : in WIND_DIRECTION);
     entry UPDATE_WIND_SPEED(WINDSPD : in WIND_SPEED);
     entry UPDATE_REL_HUM(RELHUM : in RELATIVE_HUMIDITY);
     entry UPDATE_TEMPERATURE(TEMP : in TEMPERATURE);
     entry UPDATE_BAR_PRESS(BARPRESS : in
        BAROMETRIC_PRESSURE);
     entry UPDATE_PRECIP(PRECIP : in PRECIPITATION);
   end ENVIRONMENT;
     task THREAT is
        entry ADD(HOSTILE : in HOSTILE_NAME;
          NATION : in NATIONALITY;
```

```
            SHIP : in SHIP_CLASS;
            WEPS : in WEAPONS;
            ECM : in ECM_EQUIPMENT;
            METHOD : in ENGAGEMENT_METHOD);
         entry DELETE(HOSTILE : in HOSTILE_NAME;
            NATION : in NATIONALITY);
         entry UPDATE_CLASS(HOSTILE : in HOSTILE_NAME;
            NATION : in NATIONALITY;
            SHIP : in SHIP_CLASS);
         entry UPDATE_WEPS(HOSTILE : in HOSTILE_NAME;
            NATION : in NATIONALITY;
            WEPS : in WEAPONS);
         entry UPDATE_ECM(HOSTILE : in HOSTILE_NAME;
            NATION : in NATIONALITY;
            ECM : in ECM_EQUIPMENT);
         entry UPDATE_METHOD(HOSTILE : in HOSTILE_NAME;
            NATION : in NATIONALITY;
            METHOD : in ENGAGEMENT_METHOD);
      end THREAT;
end UPDATE;
package body UPDATE is
   task body LAUNCHER_MISSILE_STATUS is
   begin
      accept UPDATE_LAU(LAUNCHER_NUMBER : in LAU_NBR;
         CANNISTER_NUMBER : in CAN_NBR;
         MISSILE_type : in MISS_type;
         LAUNCHER_INHIBIT : in LAU_INHIBIT) do
      -- insert additional body of UPDATE_LAU
      end UPDATE_LAU;
   end LAUNCHER_MISSILE_STATUS;
   task body SHIP_PARAMETER is
   begin
      accept UPDATE_COURSE(SHIP_COURSE : in COURSE) do
      -- insert additional body of UPDATE_COURSE
      end UPDATE_COURSE;
      accept UPDATE_SPEED(SHIP_SPEED : in SPEED) do
      -- insert additional body of UPDATE_SPEED
      end UPDATE_SPEED;
      accept UPDATE_LATITUDE(SHIP_LATITUDE : in LATITUDE) do
      -- insert additional body of UPDATE_LATITUDE
      end UPDATE_LATITUDE;
      accept UPDATE_LONGITUDE(SHIP_LONGITUDE : in LONGITUDE)
         do
      -- insert additional body of UPDATE_LONGITUDE
      end UPDATE_LONGITUDE;
   -- insert additional body of task SHIP_PARAMETER
   end SHIP_PARAMETER;
   task body ENVIRONMENT is
   begin
      accept UPDATE_VISIBILITY(VIS : in VISIBILITY) do
      -- insert additional body of UPDATE_VISIBILITY
      end UPDATE_VISIBILITY;
      accept UPDATE_SEA_STATE(SEA : in SEA_STATE) do
      -- insert additional body of UPDATE_SEA_STATE
      end UPDATE_SEA_STATE;
      accept UPDATE_WIND_DIR(WINDDIR : in WIND_DIRECTION) do
      -- insert additional body of UPDATE_WIND_DIR
      end UPDATE_WIND_DIR;
      accept UPDATE_WIND_SPEED(WINDSPD : in WIND_SPEED) do
      -- insert additional body of UPDATE_WIND_SPEED
      end UPDATE_WIND_SPEED;
      accept UPDATE_REL_HUM(HUM : in RELATIVE_HUMIDITY) do
      -- insert additional body of UPDATE_REL_HUM
      end UPDATE_REL_HUM;
      accept UPDATE_TEMPERATURE(TEMP : in TEMPERATURE) do
      -- insert additional body of UPDATE_TEMPERATURE
      end UPDATE_TEMPERATURE;
      accept UPDATE_BAR_PRESS(BARPRESS : in
         BAROMETRIC_PRESSURE) do
```

```
      -- insert additional body of UPDATE_BAR_PRESS
      end UPDATE_BAR_PRESS;
      accept UPDATE_PRECIP(PRECIP : in PRECIPITATION) do
      -- insert additional body of UPDATE_PRECIP
      end UPDATE_PRECIP;
    -- insert additional body of task ENVIRONMENT
    end ENVIRONMENT;
      task body THREAT is
      begin
        accept ADD(HOSTILE : in HOSTILE_NAME;
          NATION : in NATIONALITY;
          SHIP : in SHIP_CLASS;
          WEPS : in WEAPONS;
          ECM : in ECM_EQUIPMENT;
          METHOD : in ENGAGEMENT_METHOD) do
        -- insert additional body of ADD
        end ADD;
        accept DELETE(HOSTILE : in HOSTILE_NAME;
          NATION : in NATIONALITY) do
        -- insert additional body of DELETE
        end DELETE;
        accept UPDATE_CLASS(HOSTILE : in HOSTILE_NAME;
          NATION : in NATIONALITY;
          SHIP : in SHIP_CLASS) do
        -- insert additional body of UPDATE_CLASS
        end UPDATE_CLASS;
        accept UPDATE_WEPS(HOSTILE : in HOSTILE_NAME;
          NATION : in NATIONALITY;
          WEPS : in WEAPONS) do
        -- insert additional body of UPDATE_WEPS
        end UPDATE_WEPS;
        accept UPDATE_ECM(HOSTILE : in HOSTILE_NAME;
          NATION : in NATIONALITY;
          ECM : in ECM_EQUIPMENT) do
        -- insert additional body of UPDATE_ECM
        end UPDATE_ECM;
        accept UPDATE_METHOD(HOSTILE : in HOSTILE_NAME;
          NATION : in NATIONALITY;
          METHOD : in ENGAGEMENT_METHOD) do
        -- insert additional body of UPDATE_METHOD
        end UPDATE_ECM;
    -- insert ADDITION body of task THREAT
    end THREAT;
  begin
  -- insert additional body of package UPDATE
  end UPDATE;
  package AUTO_ENGAGEMENT is
    type NBR_LAUNCHERS;
    LAU_NBR : integer range 1..NBR_LAUNCHERS;
    CAN_NBR : integer range 1..4;
    MISSILE_type : integer range 1..7;
    LAU_INHIBIT : boolean;
    HOSTILE_NAME : string(1..12);
    NATIONALITY : string(1..10);
    ENGAGEMENT_METHOD : string(1..50);
      procedure AUTO_ENGAGE(LAUNCHER_NBR : in LAU_NBR;
        CANNISTER_NBR : in CAN_NBR;
        MISS_type : in MISSILE_type;
        HOSTILE : in HOSTILE_NAME;
        NATION : in NATIONALITY;
        METHOD : out  ENGAGEMENT_METHOD);
      procedure PROB_ACQUISITION(METHOD : in out
        ENGAGEMENT_METHOD);
      procedure UNCERTAINITY_ELLIPSE(METHOD : in out
        ENGAGEMENT_METHOD);
  end AUTO_ENGAGEMENT;
  package body AUTO_ENGAGEMENT is
    procedure AUTO_ENGAGE(LAUNCHER_NBR : in LAU_NBR;
      CANNISTER_NBR : in CAN_NBR;
```

```
      MISS_type : in MISSILE_type;
      HOSTILE : in HOSTILE_NAME;
      NATION : in NATIONALITY;
      METHOD : out ENGAGEMENT_METHOD) is
    begin
    -- insert additional body of procedure AUTO_ENGAGE
    end AUTO_ENGAGE;
    procedure PROB_ACQUISITION(METHOD : in out
      ENGAGEMENT_METHOD) is
    begin
    -- insert additional body of procedure PROB_ACQUISITION
    end PROB_ACQUISITION;
    procedure UNCERTAINITY_ELLIPSE(METHOD : in out
      ENGAGEMENT_METHOD) is
    begin
    -- insert additional body of procedure
    -- UNCERTAINITY_ELLIPSE
    end UNCERTAINITY_ELLIPSE;
  begin
  -- insert additional body of package AUTO_ENGAGEMENT
  end AUTO_ENGAGEMENT;
  package MANUAL_ENGAGEMENT is
    type NBR_LAUNCHERS;
    LAU_NBR : integer range 1..NBR_LAUNCHERS;
    CAN_NBR : integer range 1..4;
    MISSILE_type : integer range 1..7;
    LAU_INHIBIT : boolean;
    ENGAGEMENT_METHOD : string(1..50);
      procedure MANUAL_ENGAGE(LAUNCHER_NBR : in LAU_NBR;
        CANNISTER_NBR : in CAN_NBR;
        MISS_type : in MISSILE_type;
        METHOD : out  ENGAGEMENT_METHOD);
  end MANUAL_ENGAGEMENT;
  package body MANUAL_ENGAGEMENT is
    procedure MANUAL_ENGAGE(LAUNCHER_NBR : in LAU_NBR;
      CANNISTER_NBR : in CAN_NBR;
      MISS_type : in MISSILE_type;
      METHOD : out ENGAGEMENT_METHOD) is
    begin
    -- insert additional body of procedure MANUAL_ENGAGE
    end MANUAL_ENGAGE;
  begin
  -- insert additional body of package MANUAL_ENGAGEMENT
  end MANUAL_ENGAGEMENT;
  package DISPLAY is
    type MENU_type;
    type MAXSPEED;
    VISIBILITY : integer range 0..30;
    SEA_STATE : integer range 0..10;
    WIND_DIRECTION : integer range 0..359;
    WIND_SPEED : integer range 0..100;
    RELATIVE_HUMIDITY : integer range 0..100;
    TEMPERATURE : integer range -100..130;
    BAROMETRIC_PRESSURE : integer range 900..1100;
    type PRECIPITATION is (YES,NO);
    COURSE : integer range 0..359;
    SPEED : integer range 0..MAXSPEED;
    type LATITUDE is record
        DEGREES : integer range -90..90;
        MINUTES : integer range 0..60;
        SECONDS : integer range 0..60;
      end record;
    type LONGITUDE is record
        DEGREES : integer range -180..180;
        MINUTES : integer range 0..60;
        SECONDS : integer range 0..60;
      end record;
    HOSTILE_NAME : string(1..12);
    NATIONALITY : string(1..10);
```

```
SHIP CLASS : string(1..9);
WEAPONS : string(1..50);
ECM_EQUIPMENT : string(1..50);
ENGAGEMENT_METHOD : string(1..50);
TRACK NUMBER : integer range 100..3199;
TRACK type : integer range 1..7;
VESSEL CLASS : integer range 0..1;
type MAXrange;
HOSrange : integer range 0..MAXrange;
BEARING : integer range 0..359;
HOSTILE LAT : LATITUDE;
HOSTILE LONG : LONGITUDE;
ENGAGEMENT PLAN : string(1..50);
task MENU DISPLAY is
  entry ACCESS MENU(MENU : out MENU_type);
end MENU DISPLAY;
task LAUNCHER_MISSILE_STATUS is
  entry ACCESS_LM(LAUNCHER_NBR : out LAU_NBR;
    CANNISTER_NBR : out CAN_NBR;
    MISS type : out MISSILE type;
    INHIBIT : out LAU_INHIBIT);
end LAUNCHER MISSILE_STATUS;
task ENVIRONMENT is
  entry ACCESS_ENV(VIS : out VISIBILITY;
    SEA : out SEA_STATE;
    WINDDIR : out WIND_DIRECTION;
    WINDSPD : out WIND_SPEED;
    RELHUM : out RELATIVE_HUMIDITY;
    TEMP : out TEMPERATURE;
    BARPRESS : out BAROMETRIC PRESSURE;
    PRECIP : out PRECIPITATION);
end ENVIRONMENT;
task SHIP PARAMETER is
  entry ACCESS_SP(SHIP COURSE : out COURSE;
    SHIP SPEED : out SPEED;
    SHIP LATITUDE : out LATITUDE;
    SHIP LONGITUDE : out LONGITUDE);
end SHIP PARAMETER;
task TRACK is
  entry ACCESS_TR(TRACK_NUM : out TRACK_NUMBER;
    TRACK TYP : out TRACK_type;
    CLASS : out VESSEL_CLASS;
    RNG : out HOSrange;
    BRNG : out BEARING;
    HOST LAT : out HOSTILE_LAT;
    HOST LONG : out HOSTILE_LONG);
end TRACK;
task THREAT is
  entry ACCESS_TH(HOSTILE : out HOSTILE_NAME;
    NATION : out NATIONALITY;
    SHIP : out SHIP CLASS;
    WEPS : out WEAPONS;
    ECM : out ECM_EQUIPMENT;
    METHOD : out ENGAGEMENT_METHOD);
end THREAT;
task ENGAGEMENT PLAN is
  entry ACCESS EN(TRACK_NUM : out TRACK_NUMBER;
    PLAN : out ENGAGEMENT_PLAN);
end ENGAGEMENT PLAN;
package ENGAGEMENT DISPLAY is
  procedure AUTO_DISPLAY(TRACK_NUM : in out TRACK_NUMBER;
    PLAN : in out ENGAGEMENT PLAN;
      HOSTILE : in out HOSTILE_NAME;
      NATION : in out NATIONALITY;
      SHIP : in out SHIP CLASS;
      WEPS : in out WEAPONS;
      ECM : in out ECM EQUIPMENT;
      METHOD : in out ENGAGEMENT METHOD);
  procedure MANUAL_DISPLAY(TRACK_NUM : in out
```

```
        TRACK NUMBER;
        PLAN : in out ENGAGEMENT_PLAN;
        HOSTILE : in out HOSTILE_NAME;
        NATION : in out NATIONALITY;
        WEPS : in out WEAPONS;
        ECM : in out ECM_EQUIPMENT;
        METHOD : in out ENGAGEMENT_METHOD);
      procedure GRAPHICS;
    end ENGAGEMENT_DISPLAY;
end DISPLAY;
package body DISPLAY is
    task body MENU_DISPLAY is
    begin
      accept ACCESS_MENU(MENU : out MENU_type) do
      -- insert additional body of ACCESS_MENU
      end ACCESS_MENU;
    -- insert additional body of task MENU_DISPLAY
    end MENU_DISPLAY;
    task body LAUNCHER_MISSILE_STATUS is
    begin
      accept ACCESS_LM(LAUNCHER_NBR : out LAU_NBR;
        CANNISTER_NBR : out CAN_NBR;
        MISS_type : out MISSILE_type;
        INHIBIT : out LAU_INHIBIT) do
      -- insert additional body of ACCESS_LM
      end ACCESS_LM;
    -- insert additional body of task LAUNCHER_MISSILE_STATUS
    end LAUNCHER_MISSILE_STATUS;
    task body ENVIRONMENT is
    begin
      accept ACCESS_ENV(VIS : out VISIBILITY;
        SEA : out SEA_STATE;
        WINDDIR : out WIND_DIRECTION;
        WINDSPD : out WIND_SPEED;
        RELHUM : out RELATIVE_HUMIDITY;
        TEMP : out TEMPERATURE;
        BARPRESS : out BAROMETRIC_PRESSURE;
        PRECIP : out PRECIPITATION) do
      -- insert additional body of ACCESS_ENV
      end ACCESS_EN;
    -- insert additional body of task ENVIRONMENT
    end ENVIRONMENT;
    task body SHIP_PARAMETER is
    begin
      accept ACCESS_SP(SHIP_COURSE : out COURSE;
        SHIP_SPEED : out SPEED;
        SHIP_LATITUDE : out LATITUDE;
        SHIP_LONGITUDE : out LONGITUDE) do
      -- insert additional body of ACCESS_SP
      end ACCESS_SP;
    -- insert additional body of task SHIP_PARAMETER
    end SHIP_PARAMETER;
    task body TRACK is
    begin
      accept ACCESS_TR(TRACK_NUM : out TRACK_NUMBER;
        TRACK_TYP : out TRACK_type;
        CLASS : out VESSEL_CLASS;
        RNG : out HOSrange;
        BRNG : out BEARING;
        HOST_LAT : out HOSTILE_LAT;
        HOST_LONG : out HOSTILE_LONG) do
      -- insert additional body of ACCESS_TR
      end ACCESS_TR;
    -- insert additional body of task TRACK
    end TRACK;
    task body THREAT is
    begin
      accept ACCESS_TH(HOSTILE : out HOSTILE_NAME;
        NATION : out NATIONALITY;
```

```
        SHIP : out SHIP_CLASS;
        WEPS : out WEAPONS;
        ECM : out ECM_EQUIPMENT;
        METHOD : out ENGAGEMENT_METHOD) do
      -- insert additional body of ACCESS_TH
      end ACCESS_TH;
    -- insert additional body of task THREAT
    end THREAT;
    task body ENGAGEMENT_PLAN is
    begin
      accept ACCESS_EN(TRACK_NUM : out TRACK_NUMBER;
        PLAN : out ENGAGEMENT_PLAN) do
      -- insert additional body of ACCESS_EN
      end ACCESS_EN;
    -- insert additional body of task ENGAGEMENT_PLAN
    end ENGAGEMENT_PLAN;
    package body ENGAGEMENT_DISPLAY is
      procedure AUTO_DISPLAY(TRACK_NUM : in out TRACK_NUMBER;
        PLAN : in out ENGAGEMENT_PLAN;
        HOSTILE : in out HOSTILE_NAME;
        NATION : in out NATIONALITY;
        SHIP : in out SHIP_CLASS;
        WEPS : in out WEAPONS;
        ECM : in out ECM_EQUIPMENT;
        METHOD : in out ENGAGEMENT_METHOD) is
      begin
      -- insert additional body of procedure AUTO_DISPLAY
      end AUTO_DISPLAY;
      procedure MANUAL_DISPLAY(TRACK_NUM : in out
        TRACK_NUMBER;
        PLAN : in out ENGAGEMENT_PLAN;
        HOSTILE : in out HOSTILE_NAME;
        NATION : in out NATIONALITY;
        WEPS : in out WEAPONS;
        ECM : in out ECM_EQUIPMENT;
        METHOD : in out ENGAGEMENT_METHOD) is
      begin
      -- insert additional body of procedure MANUAL_DISPLAY
      end MANUAL_DISPLAY;
      procedure GRAPHICS is
      begin
      -- insert additional body of procedure GRAPHICS
      end GRAPHICS;
    begin
    -- insert additional body of package ENGAGEMENT_DISPLAY
    end ENGAGEMENT_DISPLAY;
  begin
  -- insert additional body of package DISPLAY
  end DISPLAY;
  package DATA_BASE_MANAGER is
    package LM_STATUS_MANAGER is
      type NBR_LAUNCHERS;
      LAU_NBR : integer range 1..NBR_LAUNCHERS;
      CAN_NBR : integer range 1..4;
      MISS_type : integer range 1..7;
      LAU_INHIBIT : boolean;
        task LAUNCHER_MISSILE_STATUS is
          entry UPDATE_LAU(LAUNCHER : in LAU_NBR;
            CANNISTER : in CAN_NBR;
            MISSILE : in MISS_type;
            LAUNCHER_INHIBIT : in  LAU_INHIBIT);
          entry ACCESS_LM(LAUNCHER : out LAU_NBR;
            CANNISTER : out CAN_NBR;
            MISSILE : out MISS_type;
            LAUNCHER_INHIBIT : out LAU_INHIBIT);
        end LAUNCHER_MISSILE_STATUS;
    end LM_STATUS_MANAGER;
    package SP_MANAGER is
      type MAXSPEED;
```

```
    COURSE : integer range 0..359;
    SPEED : integer range 0..MAXSPEED;
    type LATITUDE is record
        DEGREES : integer range -90..90;
        MINUTES : integer range 0..60;
        SECONDS : integer range 0..60;
      end record;
    type LONGITUDE is record
        DEGREES : integer range -180..180;
        MINUTES : integer range 0..60;
        SECONDS : integer range 0..60;
      end record;
      task SHIP_PARAMETER is
        entry UPDATE_SP(SHIP_COURSE : in COURSE;
          SHIP_SPEED : in SPEED;
          SHIP_LATITUDE : in LATITUDE;
          SHIP_LONGITUDE : in LONGITUDE);
        entry ACCESS_SP(SHIP_COURSE : out COURSE;
          SHIP_SPEED : out SPEED;
          SHIP_LATITUDE : out LATITUDE;
          SHIP_LONGITUDE : out LONGITUDE);
      end SHIP_PARAMETER;
    end SP_MANAGER;
package ENVIRONMENT_MANAGER is
    VISIBILITY : integer range 0..30;
    SEA_STATE : integer range 0..10;
    WIND_DIRECTION : integer range 0..359;
    WIND_SPEED : integer range 0..100;
    RELATIVE_HUMIDITY : integer range 0..100;
    TEMPERATURE : integer range -100..130;
    BAROMETRIC_PRESSURE : integer range 900..1000;
    PRECIPITATION : boolean;
      task ENVIRONMENT is
        entry UPDATE_ENV(VIS : in VISIBILITY;
          SEA : in SEA_STATE;
          WINDDIR : in WIND_DIRECTION;
          WINDSPD : in WIND_SPEED;
          RELHUM : in RELATIVE_HUMIDITY;
          TEMP : in TEMPERATURE;
          BARPRESS : in BAROMETRIC_PRESSURE;
          PRECIP : in PRECIPITATION);
        entry ACCESS_ENV(VIS : out VISIBILITY;
          SEA : out SEA_STATE;
          WINDDIR : out WIND_DIRECTION;
          WINDSPD : out WIND_SPEED;
          RELHUM : out RELATIVE_HUMIDITY;
          TEMP : out TEMPERATURE;
          BARPRESS : out BAROMETRIC_PRESSURE;
          PRECIP : out PRECIPITATION);
      end ENVIRONMENT;
end ENVIRONMENT_MANAGER;
package THREAT_MANAGER is
    HOSTILE_NAME : string(1..12);
    NATIONALITY : string(1..10);
    SHIP_CLASS : string(1..9);
    WEAPONS : string(1..50);
    ECM_EQUIPMENT : string(1..50);
    ENGAGEMENT_METHOD : string(1..50);
      task THREAT is
        entry ACCESS_TH(HOSTILE : out HOSTILE_NAME;
          NATION : out NATIONALITY;
          SHIP : out SHIP_CLASS;
          WEPS : out WEAPONS;
          ECM : out ECM_EQUIPMENT;
          METHOD : out ENGAGEMENT_METHOD);
      end THREAT;
end THREAT_MANAGER;
package TRACK_MANAGER is
    TRACK_NUMBER : integer range 100..3199;
```

```
    TRACK_type : integer range 1..7;
    VESSEL_CLASS : integer range 0..1;
    type MAXrange;
    HCSrange : integer range 0..MAXrange;
    BEARING : integer range 0..359;
    type HOSTILE_LAT is record
        DEGREES : integer range -90..90;
        MINUTES : integer range 0..60;
        SECONDS : integer range 0..60;
      end record;
    type HOSTILE_LONG is record
        DEGREES : integer range -180..180;
        MINUTES : integer range 0..60;
        SECONDS : integer range 0..60;
      end record;
      task TRACK is
        entry ADD(TRACK_NUM : in TRACK_NUMBER;
          TRACK_TYP : in TRACK_type;
          CLASS : in VESSEL_CLASS;
          RNG : in HCSrange;
          BRNG : in BEARING;
          HOST_LAT : in HOSTILE_LAT;
          HOST_LONG : in HOSTILE_LONG);
        entry DELETE(TRACK_NUM : in TRACK_NUMBER;
          TRACK_TYP : in TRACK_type);
        entry ACCESS_TR(TRACK_NUM : out TRACK_NUMBER;
          TRACK_TYP : out TRACK_type;
          CLASS : out VESSEL_CLASS;
          RNG : out HCSrange;
          BRNG : out BEARING;
          HOST_LAT : out HOSTILE_LAT;
          HOST_LONG : out HOSTILE_LONG);
        entry UPDATE_TY(TRACK_NUM : in TRACK_NUMBER;
          TRACK_TYP : in TRACK_type);
        entry UPDATE_CL(TRACK_NUM : in TRACK_NUMBER;
          CLASS : in VESSEL_CLASS);
        entry UPDATE_RN(TRACK_NUM : in TRACK_NUMBER;
          RNG : in HCSrange);
        entry UPDATE_BR(TRACK_NUM : in TRACK_NUMBER;
          BRNG : in BEARING);
        entry UPDATE_LA(TRACK_NUM : in TRACK_NUMBER;
          HOST_LAT : in HOSTILE_LAT);
        entry UPDATE_LO(TRACK_NUM : in TRACK_NUMBER;
          HOST_LONG : in HOSTILE_LONG);
    end TRACK;
  end TRACK_MANAGER;
  package MENU_MANAGER is
    type MENU_type;
      task MENU is
        entry ACCESS_ME(MENU_TY : out MENU_type);
      end MENU;
  end MENU_MANAGER;
  package ENGAGEMENT_PLAN_MANAGER is
    TRACK_NUMBER : integer range 100..3199;
    ENGAGEMENT_PLAN : string(1..50);
    HOSTILE_NAME : string(1..12);
    NATIONALITY : string(1..10);
    SHIP_CLASS : string(1..9);
    WEAPONS : string(1..50);
    ECM_EQUIPMENT : string(1..50);
    ENGAGEMENT_METHOD : string(1..50);
      task ENGAGEMENT_PLAN is
        entry ACCESS_EN(TRACK_NUM : out TRACK_NUMBER;
          PLAN : out ENGAGEMENT_PLAN;
          HOSTILE : out HOSTILE_NAME;
          NATION : out NATIONALITY;
          SHIP : out SHIP_CLASS;
          WEPS : out WEAPONS;
          ECM : out ECM_EQUIPMENT;
```

```
            METHOD : out ENGAGEMENT_METHOD);
        end ENGAGEMENT_PLAN;
    end ENGAGEMENT_PLAN_MANAGER;
end DATA_BASE_MANAGER;
package body DATA_BASE_MANAGER is
    package body LM_STATUS_MANAGER is
        task body LAUNCHER_MISSILE_STATUS is
        begin
          accept UPDATE_LAU(LAUNCHER : in LAU_NBR;
            CANNISTER : in CAN_NBR;
            MISSILE : in MISS_type;
            LAUNCHER_INHIBIT : in  LAU_INHIBIT) do
          -- insert additional body of UPDATE_LAU
          end UPDATE_LAU;
          accept ACCESS_LM(LAUNCHER : out  LAU_NBR;
            CANNISTER : out CAN_NBR;
            MISSILE : out MISS_type;
            LAUNCHER_INHIBIT : out LAU_INHIBIT) do
          -- insert additional body of ACCESS_LM;
          end ACCESS_LM;
        end LAUNCHER_MISSILE_STATUS;
    begin
    -- insert additional body of package LM_STATUS_MANAGER
    end LM_STATUS_MANAGER;
    package body SP_MANAGER is
      task body SHIP_PARAMETER is
      begin
        accept UPDATE_SP(SHIP_COURSE : in COURSE;
          SHIP_SPEED : in SPEED;
          SHIP_LATITUDE : in LATITUDE;
          SHIP_LONGITUDE : in LONGITUDE) do
        -- insert additional body of UPDATE_SP
        end UPDATE_SP;
        accept ACCESS_SP(SHIP_COURSE : out COURSE;
          SHIP_SPEED : out SPEED;
          SHIP_LATITUDE : out LATITUDE;
          SHIP_LONGITUDE : out LONGITUDE) do
        -- insert additional body of ACCESS_SP
        end ACCESS_SP;
      end SHIP_PARAMETER;
    begin
    -- insert additional body of package SP_MANAGER
    end SP_MANAGER;
    package body ENVIRONMENT_MANAGER is
        task body ENVIRONMENT is
        begin
          accept UPDATE_ENV(VIS : in VISIBILITY;
            SEA : in SEA_STATE;
            WINDDIR : in WIND_DIRECTION;
            WINDSPD : in WIND_SPEED;
            RELHUM : in RELATIVE_HUMIDITY;
            TEMP : in TEMPERATURE;
            BARPRESS : in BAROMETRIC_PRESSURE;
             PRECIP : in PRECIPITATION) do
          -- insert additional body of UPDATE_ENV
          end UPDATE_ENV;
          accept ACCESS_ENV(VIS : out VISIBILITY;
            SEA : out SEA_STATE;
              WINDDIR : out WIND_DIRECTION;
              WINDSPD : out WIND_SPEED;
              RELHUM : out RELATIVE_HUMIDITY;
              TEMP : out TEMPERATURE;
              BARPRESS : out BAROMETRIC_PRESSURE;
              PRECIP : out PRECIPITATION) do
          -- insert additional body of ACCESS_ENV
          end ACCESS_ENV;
        end ENVIRONMENT;
    begin
    -- insert additional body of package ENVIRONMENT_MANAGER
```

```
end ENVIRONMENT_MANAGER;
package body THREAT_MANAGER is
    task body THREAT is
    begin
      accept ACCESS_TH(HOSTILE : out HOSTILE_NAME;
        NATION : out NATIONALITY;
        SHIP : out SHIP_CLASS;
           WEPS : out WEAPONS;
           ECM : out ECM_EQUIPMENT;
           METHOD : out ENGAGEMENT_METHOD) do
      -- insert additional body of ACCESS_TH
      end ACCESS_TH;
    end THREAT;
begin
-- insert additional body of package THREAT_MANAGER
end THREAT_MANAGER;
package body TRACK_MANAGER is
    task body TRACK is
    begin
      accept ADD(TRACK_NUM : in TRACK_NUMBER;
        TRACK_TYP : in TRACK_type;
        CLASS : in VESSEL_CLASS;
        RNG : in HOSrange;
        BRNG : in BEARING;
        HOST_LAT : in HOSTILE_LAT;
        HOST_LONG : in HOSTILE_LONG) do
      -- insert additional body of ADD;
      end ADD;
      accept DELETE(TRACK_NUM : in TRACK_NUMBER;
        TRACK_TYP : in TRACK_type) do
      -- insert additional body of DELETE
      end DELETE;
      accept ACCESS_TR(TRACK_NUM : out TRACK_NUMBER;
        TRACK_TYP : out TRACK_type;
         CLASS : out VESSEL_CLASS;
         RNG : out HOSrange;
         BRNG : out BEARING;
         HOST_LAT : out HOSTILE_LAT;
         HOST_LONG : out HOSTILE_LONG) do
       -- insert additional body of ACCESS_TR
       end ACCESS_TR;
       accept UPDATE_TY(TRACK_NUM : in TRACK_NUMBER;
         TRACK_TYP : in TRACK_type) do
       -- insert additional body of UPDATE_TY
       end UPDATE_TY;
       accept UPDATE_CL(TRACK_NUM : in TRACK_NUMBER;
         CLASS : in VESSEL_CLASS) do
       -- insert additional body of UPDATE_CL
       end UPDATE_CL;
       accept UPDATE_RN(TRACK_NUM : in TRACK_NUMBER;
         RNG : in HOSrange) do
       -- insert additional body of UPDATE_RN
       end UPDATE_RN;
       accept UPDATE_BR(TRACK_NUM : in TRACK_NUMBER;
         BRNG : in BEARING) do
       -- insert additional body of UPDATE_TR
       end UPDATE_TR;
       accept UPDATE_LA(TRACK_NUM : in TRACK_NUMBER;
         HOST_LAT : in HOSTILE_LAT) do
       -- insert additional body of UPDATE_LA
       end UPDATE_LA;
       accept UPDATE_LO(TRACK_NUM : in TRACK_NUMBER;
         HOST_LONG : in HOSTILE_LONG) do
       -- insert additional body of UPDATE_LO
       end UPDATE_LO;
  end TRACK;
begin
-- insert additional body of package TRACK_MANAGER
end TRACK_MANAGER;
```

```
package body MENU_MANAGER is
     task body MENU is
     begin
       accept  ACCESS_ME(MENU_TY : out MENU_type) do
       -- insert additional body of MENU
       end ACCESS_ME;
     end MENU;
begin
-- insert additional body of package MENU_MANAGER
end MENU_MANAGER;
package body ENGAGEMENT_PLAN_MANAGER is
     task body ENGAGEMENT_PLAN is
     begin
       accept ACCESS_EN(TRACK_NUM : out TRACK_NUMBER;
         PLAN : out ENGAGEMENT_PLAN;
         HOSTILE : out HOSTILE_NAME;
         NATION : out NATIONALITY;
         SHIP : out SHIP_CLASS;
         WEPS : out WEAPONS;
         ECM : out ECM_EQUIPMENT;
         METHOD : out ENGAGEMENT_METHOD) do
        -- insert additional body of ACCESS_EN
        end ACCESS_EN;
     end ENGAGEMENT_PLAN;
begin
-- insert additional body of package
-- ENGAGEMENT_PLAN_MANAGER
end ENGAGEMENT_PLAN_MANAGER;
begin
-- insert additional body of package DATA_BASE_MANAGER
end DATA_BASE_MANAGER;
```

LIST OF REFERENCES

1. Naval Sea Systems Command, XWS-20842A, Performance Design Requirements for HARPOON Ship Command-Launch Control Set, AN/SWG-1A(V), 28 January 1983.

2. Sentman, Lawrence and Maroney, Randal, Integrated Design Specifications for the HARPOON Shipboard Command Launch Control Set, Master's Thesis, Naval Postgraduate School, Monterey, California, December, 1982.

3. Olivier, Daniel P. and Olsen, Kevin R., A Design Methodology for Embeded Weapons Systems Using the HARPOON Shipboard Command-Launch Control Set (HSCLCS) AN/SWG-1A(V), Master's Thesis, Naval Postgraduate School, Monterey, California, June, 1983.

4. Ransbotham, James I. and Moorehead, Donald F., A Program Manager's Methodology for Developing Structured Design in Embedded Weapons Systems, Master's Thesis, Naval Postgraduate School, Monterey, California, December 1983.

BIBLIOGRAPHY

Barnes, J. G. P., Programming in Ada, Addison-Wesley Publishing Company, 1982.

Shannon, Robert E., System Simulation: The Art and Science, Prentice-Hall, Inc., N.J., 1975.

Law, Averill M. and Kelton, W. David, Simulation Modeling and Analysis, McGraw-Hill Book Company, N.Y., 1982.

Spriet, Jan A. and Vansteenbiste, Ghislain C., Computer-Aided Modelling and Simulation, Academic Press, London, 1982.

Gordon, Geoffrey, System Simulation, Prentice-Hall, Inc., N.J., 1978.

DeMarco, Tom, Structured Analysis and System Specification, Prentice-Hall, Inc., N.J., 1979.

DeMarco, Tom, Controlling Software Projects: Management Measurement & Estimation, Yourdon Press, N.Y., 1982.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943	2
3.	Department Chairman, Code 52 Department of Computer Science Naval Postgraduate School Monterey, California 93943	1
4.	LT Judd R. Eschliman 818 Lucerne Avenue Pensacola, Florida 32505	1
5.	NSWSES Code 4613 (Mr. Paul Gognon) Port Hueneme, California 93043	1
6.	Naval Sea Systems Command Department of the Navy ATTN: Mr. Lee Minin NAVSEA Code SEA-62WB Washington, D.C. 20362	1
7.	Mr. Fred S. Gais Director - HARPOON Ship Integration McDonnell Douglas Astronautics Company St. Louis Division P.O. Box 516 St. Louis, Missouri 63166	1
8.	LCDR Ronald Modes, USN, VAQ 129 NAS Whidbey Island, Washington 98278	1
9.	LCDR Ronald Kurth, USN, Code 52KH Department of Computer Science Naval Postgraduate School Monterey, California 93943	2
10.	Computer Technologies Curricular Office Naval Postgraduate School Code 37 Monterey, California 93943	1

END

FILMED

4-84

DTIC